Rafał Adamczak

Zastosowanie sieci neuronowych do klasyfikacji danych doświadczalnych.

Praca doktorska
pod kierunkiem prof. Włodzisława Ducha

Katedra Metod Komputerowych
Uniwersytet Mikołaja Kopernika
2001

1. Wprowadzenie

Sztuczne sieci neuronowe powstały na bazie inspiracji neurobiologicznych. Zbudowane są z połączonych ze sobą sztucznych neuronów, które jednak z punktu widzenia neurobiologii są skrajnie uproszczonym modelem rzeczywistego neuronu. Mimo to występuje szereg analogii zarówno w budowie neuronu jak i struktur, które są za pomocą tego elementu tworzone. Skrajne uproszczenie modelu neuronu nie powoduje jednak zniszczenia najważniejszej cechy występującej w rzeczywistych sieciach neuronowych, czyli zdolności uczenia się. Obecnie sieci neuronowe są narzędziem pozwalającym dokonać odwzorowania bardzo złożonych funkcji. W porównaniu z dotychczas istniejącymi metodami modelowania liniowego, sieci neuronowe mają dużo większe możliwości z powodu swojego nieliniowego charakteru.

Sieci neuronowe tworzą dziedzinę będąca na pograniczu statystyki, fizyki oraz neurobiologii. Wszystkie te dziedziny wnoszą swój wkład zarówno do rozwoju nowych algorytmów, typów sieci, jak również poprzez użycie istniejących teorii w tych dziedzinach, pozwalają lepiej opisać działanie sieci. Przykładem takiego wpływu jest zastosowanie wywodzącej się z fizyki statystycznej teorii szkieł spinowych do opisu sieci Hopfielda [29]. Podobnie uczenie sieci zwanej maszyną Boltzmanna oparte jest na metodach mechaniki statystycznej [29].

W ostatnich latach można było zaobserwować eksplozję badań algorytmów uczących się wywodzących się ze sztucznej inteligencji oraz sieci neuronowych. Rozwój ten zaowocował licznymi zastosowaniami tych algorytmów w wielu dziedzinach nauki, w tym również w fizyce, medycynie i chemii. Trudno jest wymienić wszystkie istniejące zastosowania sieci neuronowych, dlatego też w celu zobrazowania możliwości zastosowania sieci neuronowych przedstawię tylko kilka z nich i tylko te, które dotyczą klasyfikacji.

Sieci neuronowe w zastosowaniu do fizyki wysokich energii (HEP-High Energy Physics) udowodniły, że są bardziej wydajnymi klasyfikatorami od dotychczas stosowanych metod dyskryminacyjnych. Zostały zastosowane z powodzeniem zarówno w problemach czasu rzeczywistego jak i w obróbce danych tzw. „off-line". Nowe doświadczenia w dziedzinie HEP dotyczą często rzadkich procesów fizycznych. Jednym z takich procesów jest powstawanie cząstek Higgsa podczas zderzenia dwóch protonów [14]. Prawdopodobieństwo powstawania cząstki Higgsa jest tak małe, że zdarzenie musi nastąpić 10^{34} razy na sekundę po to by wykryć rozsądną liczbę cząstek Higgsa, rzędu 1000, w ciągu jednego roku. Prawdopodobieństwo innych procesów natomiast jest wyższe o czynnik 10^{13}. To oznacza, że podczas jednego roku pracy, wygenerowanych zostanie około 10^9 na sekundę zdarzeń będących tłem. Do oddzielenia tła od interesujących zdarzeń stosuje się elektroniczne urządzenia zwane dyskryminatorami. Do tego celu doskonale nadają się sieci neuronowe, które z uwagi na możliwą równoległą architekturę są bardzo szybkie.

Drugim ważnym zastosowaniem sieci neuronowych w HEP jest analiza „off-line". W przeciwieństwie do poprzednio opisanej nie ma tutaj w ogóle ograniczeń czasowych. Kluczowym zagadnieniem zastosowania sieci neuronowych w tego typu analizie jest opracowanie efektywnych metod dyskryminacji pozwalających oddzielić rzadkie interesujące procesy od procesów tła. Jak się okazuje stosowane dotychczas metody jednowymiarowych dyskryminacji są mniej efektywne od dyskryminacji wielowymiarowych.

Duże znaczenia ma również zastosowanie algorytmów uczących się w dziedzinach, w których zastosowanie istniejących teorii byłoby zbyt kosztowne, lub też, dla których nie istnieją opisujące je teorie. Przykładem takiego zastosowania są problemy SAR (Structure Activity

Relationships) lub QSAR (Quantitative SAR) [56], [32], w których w wielu przypadkach na podstawie teorii kwantowych nie da się określić związku pomiędzy aktywnością złożonych molekuł a ich strukturą. W tego typu problemach celem jest stworzenie predyktywnej teorii (np. w postaci reguł logicznych) w oparciu o analizę zbioru związków chemicznych o znanej strukturze i aktywności. Związki chemiczne opisywane są poprzez dużą liczbę cech wynikających z ich struktury molekularnej, takich jak: deskryptory kwantowo-chemiczne, deskryptory topologiczne itp.

Zrozumienie relacji pomiędzy aktywnością a strukturą związków chemicznych pozwala podjąć próbę zbudowania odpowiedniej teorii oraz znacznie przyspieszyć proces tworzenia nowych lekarstw.

Innym bardzo ważnym zastosowaniem algorytmów uczących się jest chromatografia jonowa (IC, ion chromatography) [42],[51],[43]. Zastosowanie IC wymaga dużej wiedzy z chemii jonów oraz metod chromatograficznych. Różne mechanizmy chromatografii mogą być zastosowane do separacji jonów a każda z nich wymaga rozważenia różnych warunków. Do rozwiązania tych problemów próbuje się stworzyć system ekspertowy, oparty na regułach wyznaczonych za pomocą algorytmu uczącego się.

Chociaż istnieje wiele programów realizujących różne wersje modeli neuronowych, nie są one pozbawione wad. W pierwszym rzędzie wymienić należy trudności interpretacyjne sposobu działania większości sieci. Konieczność wyboru wielu parametrów (liczba neuronów, architektura sieci i parametry wpływające na uczenie), utrudnia stosowanie takich modeli w praktyce. Celem mojej pracy był rozwój nowych algorytmów neuronowych pozbawionych w znacznej mierze takich wad. Model MLP2LN jest modyfikacją najbardziej popularnych sieci typu MLP (Multilayer Perceptron, czyli perceptronów wielowarstwowych), przekształcającą te sieci w sieci realizujące funkcje logiczne (LN, Logical Network), dzięki czemu możliwa staje się analiza danych prowadząca do reguł logicznych. Takie reguły wykorzystać można w systemach ekspertowych. Model FSM (Feature Space Mapping) jest siecią estymującą gęstość prawdopodobieństwa wektorów. Łączy on w sobie możliwości rozmytych systemów logicznych z algorytmami typu sieci neuronowych. Obydwa algorytmy przeznaczone są do problemów klasyfikacji. W końcowej części pracy przedstawione zostanie zastosowanie tych algorytmów do analizy danych doświadczalnych.

W drugim rozdziale zaprezentowane zostało zagadnienie klasyfikacji. Z zagadnieniem tym związane jest powstawanie obszarów i granic decyzyjnych oraz cel procesu uczenia. Bardzo istotnym problemem poruszonym w tym rozdziale są również metody oceny jakości klasyfikacji modeli uczących się takie jak kroswalidacja i bootstrapping. Została również omówiona stabilność klasyfikatorów.

W rozdziale trzecim przedstawiono opis metod wstępnego przetwarzania danych, których celem jest poprawienie działania klasyfikatorów np. poprzez zmniejszenie wymiarowości za pomocą różnych metod heurystycznych lub też poprzez zastosowanie specjalnych transformacji. W przypadku wielu klasyfikatorów, np. sieci neuronowych, wstępne przetwarzanie jest niekiedy niezbędne, aby móc w ogóle zastosować te klasyfikatory np. problemy wartości symbolicznych czy też wartości brakujących.

Rozdział czwarty zawiera opis algorytmów grupowania, które mogą być przydatne zarówno w problemach klasyfikacyjnych bez nadzoru jak i z nadzorem. Do inicjalizacji sieci FSM używane są metody oparte na histogramach lub dendrogramach. Pozwala to określić wstępną rozdzielczość danych, która w niektórych przypadkach może okazać się wystarczająca. Oznacza to, że po wstępnej inicjalizacji uczenie nie jest już potrzebne. Oprócz dwóch wymienionych algorytmów grupowania, przedstawiony również został jeden z najpopularniejszych, algorytm *k*-średnich.

W rozdziale piątym opisano algorytm propagacji wstecznej błędu oraz podstawowe informacje dotyczące sieci MLP. Oprócz podstawowego algorytmu uczenia sieci MLP, przedstawione zostały również najbardziej popularne odmiany tego algorytmu takie jak: Quickprop, Rprop. Do bardzo ważnych zagadnień związanych z sieciami neuronowymi należy zagadnienie regularyzacji, którego zastosowania dotyczące sieci MLP przedstawione są również w tym rozdziale.

Rozdział szósty przedstawia odmienny typ sieci w stosunku do sieci MLP, sieć RBF, w której jako funkcje transferu stosuje się funkcje radialne. Zastosowanie funkcji radialnych umożliwia tworzenie odmiennych obszarów decyzyjnych w porównaniu do tych, które tworzy sieć MLP. Przedstawiony jest podstawowy algorytm uczenia tego typu sieci. Podobnie jak w przypadku sieci MLP zaprezentowana jest konstruktywistyczna wersja sieci RBF, zwana siecią RAN, oraz sieć oparta na sieci RBF służąca do wyciągania reguł logicznych RecBFN.

W rozdziale siódmym przedstawiono krótki opis metod wyciągania reguł logicznych z sieci neuronowych, lub za pomocą sieci neuronowych. Opisano tu nowy algorytm, który został stworzony i przetestowany w Katedrze Metod Komputerowych. Ekstrakcja reguł logicznych jest od samego początku wbudowana w algorytm uczenia tej sieci. Użyta tu wersja konstruktywistyczna pozwala uniknąć problemów charakterystycznych dla sieci MLP i przyspieszyć proces uczenia. Poza tym została zaprezentowana modyfikacja sieci MLP, nieeuklidesowa sieć MLP, pozwalająca tworzyć obszary decyzyjne o różnych kształtach.

Rozdział ósmy przedstawia sieć FSM, której korzenie wywodzą się z sieci RBF, z tym, że jako funkcje transferu można w niej zastosować dowolną funkcję zlokalizowaną i faktoryzowalną. Funkcja taka nie musi być różniczkowalna, dlatego też mogą zostać użyte takie funkcje jak prostokątna czy trójkątna. Umożliwia to zwiększenie zróżnicowania tworzonych obszarów decyzyjnych przez sieć FSM. Dokładniejsze dopasowanie obszarów decyzyjnych, tworzonych przez model FSM, do danych umożliwia także wprowadzenie obrotów funkcji transferu.

Rozdział dziewiąty prezentuje zastosowanie modelu FSM oraz sieci MLP2LN do licznych danych doświadczalnych.

2. Zagadnienie klasyfikacji

Przez pojęcie klasyfikacji rozumie się dzielenie dowolnego zbioru elementów na grupy, do których zalicza się elementy różniące się, ale podobne tj. mające własności wyróżniające daną grupę. Zbiór elementów należących do jednej grupy nazywany jest klasą, a każdy element klasy - obiektem. Elementy klasy mogą różnić się między sobą, z wyjątkiem tych własności, na których opiera się klasyfikacja.

W zależności od rodzaju dostępnej informacji w ramach klasyfikacji można wyodrębnić dwa zagadnienia:

1. klasyfikację wzorcową; struktura kategorii jest znana, czyli dysponuję się charakterystyką klas, z których pochodzą obiekty; zagadnienie to nazywane jest uczeniem (rozpoznawaniem) z nauczycielem lub też pod nadzorem
2. klasyfikację bezwzorcową, znaną jako taksonomia albo analiza skupień (klasteryzacja), określaną również jako uczenie lub rozpoznawanie bez nauczyciela.

Praca ta prawie w całości poświęcona jest rozwojowi modelu FSM i MLP2LN oraz zastosowaniu obu modeli w zagadnieniach klasyfikacji pod nadzorem. Załóżmy, że dany jest zbiór przypadków (obiektów), następującej postaci:

$$\{\mathbf{x}_i, C^{(i)}\} \quad i = 1...N$$

gdzie **x** jest wektorem *n*-wymiarowym opisującym dany przypadek, natomiast $C^{(i)} \in \{C_1, C_2, \ldots, C_L\}$ etykietą klasy. Etykietą klasy może być zarówno ciąg znaków będący np. nazwą klasy jak i unikalny numer klasy. Dla odróżnienia etykieta klasy reprezentowana przez ciąg znaków oznaczana będzie przez symbol *C*, natomiast etykieta w postaci numeru klasy przez symbol *d*. Postać etykiety klasy zależy od algorytmu, który używany jest do klasyfikacji danych. Niektóre algorytmy mogą mieć etykiety w postaci nazw (np. drzewa decyzyjne), natomiast inne muszą mieć wartość numeryczną (np. większość sieci neuronowych). Poszczególne składowe wektora **x** charakteryzują dany obiekt i nazywane są cechami.

Systemy klasyfikujące (klasyfikatory), dokonują mapowania wektorów wejściowych **x** na etykietę klasy *C*. Mapowanie polega na takim doborze parametrów adaptacyjnych **W** danego modelu, aby spełniona była zależność:

$$y(\mathbf{x}_i; \mathbf{W}) = C^{(i)};\ i = 1...N \tag{1.1}$$

gdzie **W** jest zbiorem parametrów klasyfikatora.

Proces dopasowania parametrów adaptacyjnych nazywany jest uczeniem lub trenowaniem, natomiast dane, na podstawie których następuje wyznaczanie tych parametrów, danymi treningowymi.

Celem uczenia, oprócz dokładności dopasowania do danych treningowych, jest jakość generalizacji modelu, czyli jakość klasyfikacji na danych, które nie zostały użyte podczas uczenia modelu (są to tzw. dane testowe).

Uczenie klasyfikatora powinno być wykonywane tylko na próbce reprezentatywnej tzn. takiej, która spełnia poniższe warunki:

1. każdy element próbki powinien być pobierany z poszczególnych klas (populacji) w sposób losowy
2. próbka powinna być dostatecznie liczna.

Warunek 2 jest niestety trudny do sprecyzowania, gdyż jest on zależny od wielu czynników np. stopnia zaszumienia danych, stopnia nakładania się klas na siebie, liczby cech opisujących obiekty w ciągu treningowym. Można jednak na pewno stwierdzić, że im próbka jest liczniejsza, tym lepiej odzwierciedla własności poszczególnych klas.

2.1. Obszary decyzyjne

Z geometrycznego punktu widzenia każdemu wektorowi **x** z ciągu treningowego można przyporządkować punkt w odpowiednio określonej przestrzeni cech (przestrzeni obserwacji), której wymiar równy jest wymiarowi wektora treningowego.

Klasyfikator dokonuje podziału przestrzeni cech na obszary decyzyjne tzn. części przestrzeni cech, takie, że wszystkim punktom znajdującym się w tym obszarze odpowiada taka sama decyzja (klasa). Granice pomiędzy tymi obszarami nazywane są powierzchniami decyzyjnymi lub też granicami decyzji. Inaczej mówiąc granica decyzji jest to obszar w przestrzeni cech, dla którego odpowiednie funkcje kryterialne (funkcje przynależności do danej klasy) mają tą samą wartość.

Kształty obszarów decyzyjnych zależą od danych treningowych, jak również od zastosowanego modelu klasyfikacyjnego. Ze statystycznego punktu widzenia optymalna granic decyzyjna wyznaczana jest na podstawie prawdopodobieństw a posteriori. Optymalny klasyfikator wybiera klasę, dla której prawdopodobieństwo a posteriori jest większe. Dany wektor **x** klasyfikowany jest do klasy C_k wtedy, gdy spełniona jest następująca relacja:

$$p(C_k \mid \mathbf{x}) > p(C_j \mid \mathbf{x}) \quad j \neq k \tag{1.2}$$

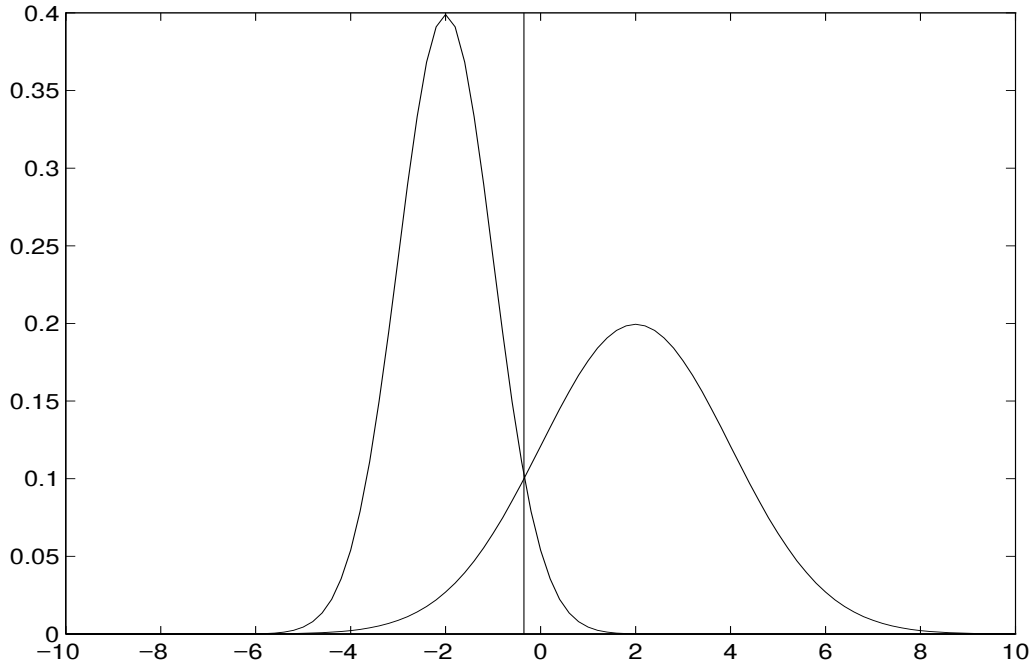
gdzie $p(C_k \mid \mathbf{x})$ jest prawdopodobieństwem a posteriori tego, że przy zadanym wzorcu **x** należy on do klasy C_k. Korzystając z twierdzenia Bayesa

$$p(C_k \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid C_k)\, p(C_k)}{p(\mathbf{x})} \tag{1.3}$$

otrzymuje się

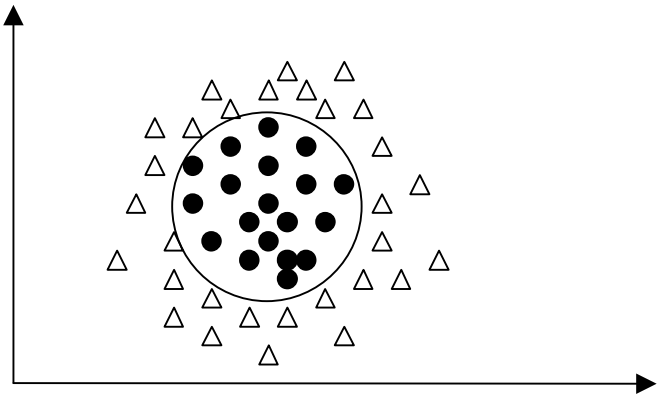
$$p(\mathbf{x} \mid C_k)\, p(C_k) > p(\mathbf{x} \mid C_j)\, p(C_j) \quad k \neq j \tag{1.4}$$

gdzie $p(C_k)$ jest prawdopodobieństwem a priori wystąpienia klasy C_k, natomiast $p(\mathbf{x}|C_k)$ jest rozkładem próbek dla klasy C_k. Równanie to prowadzi do rozwiązania, w którym granica decyzyjna między klasami znajduje się na przecięciu rozkładów tych klas. Przykład jednowymiarowy przedstawia poniższy rysunek.



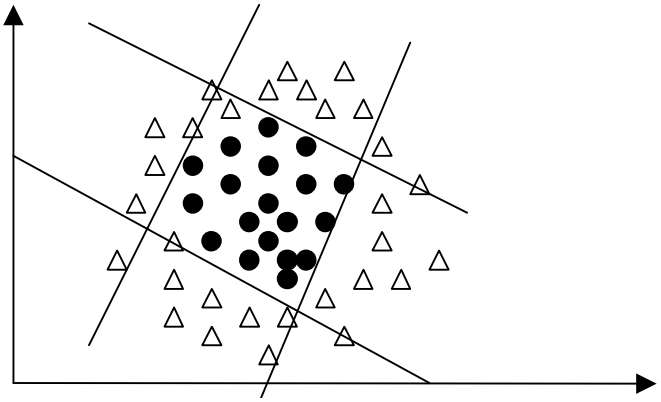
Rys 2:1 Bayesowska granica decyzyjna

Każdy z modeli klasyfikacyjnych ma jednak ograniczone możliwości tworzenia kształtów obszarów decyzyjnych, w wyniku czego czasem nie jest możliwe utworzenia optymalnej granicy decyzyjnej, lub też utworzenia takiej granicy wymaga użycia dużej liczby parametrów adaptacyjnych. Dla przykładu rozpatrzmy dane przedstawione na poniższym rysunku.



Rys 2:2 Optymalna granica decyzyjna

Najlepszą granicą decyzyjną do odseparowania tych dwóch klas jest okrąg. W przypadku jednak, gdy model tworzy obszary decyzyjne tylko za pomocą prostych, rozwiązanie, które jest on w stanie wytworzyć ma postać:



Rys 2:3 Nieoptymalna granica decyzyjna

Powstały obszar decyzyjny jest odwzorowany za pomocą czterech prostych, co powoduje, że liczba parametrów adaptacyjnych jest duża w porównaniu z modelem, który do tworzenia obszarów decyzyjnych używa elipsoid. Może to prowadzić do dłuższego czasu uczenia z uwagi na trudności w odwzorowaniu tego obszaru.

Niestety, najczęściej nie wiadomo jaki kształt mają obszary decyzyjne występujące w analizowanych danych. Dlatego też dla każdych danych stosuje się różne modele i bada ich generalizację, a za najlepszy uznawany jest ten, którego generalizacja była największa. Jeżeli generalizacja kilku modeli jest porównywalna, wybierany jest model, który spełnia jakieś inne dodatkowe kryterium np. ma najprostszą strukturę, występują możliwości wyjaśnienia przez model podejmowanych decyzji czy też jest prosty w użyciu.

2.2. Uczenie

Uczenie klasyfikatora opiera się na minimalizowaniu jakieś funkcji, zwanej funkcją błędu lub kosztu. W najprostszym przypadku tzn. gdy wszystkie błędy są jednakowo niekorzystne, funkcja taka ma postać

$$E = \sum_i Er(y(\mathbf{x}_i; \mathbf{W}), C^{(i)}) \tag{1.5}$$

gdzie funkcja *Er*

$$Er(C_i, C_j) = \begin{cases} 0, & jeżeli \quad i = j \\ 1, & jeżeli \quad i \neq j \end{cases} \tag{1.6}$$

W ogólności *Er*( ) jest funkcją macierzową określającą koszt błędnego zaklasyfikowania obiektu z klasy C_i do innej klasy np. C_j. Oto przykład takiej macierzy dla przypadku czteroklasowego

klasa/*klasa*	C_1	C_2	C_3	C_4
C_1	0	3	2	3
C_2	2	0	5	4
C_3	2	2	0	2
C_4	2	6	2	0

Jak widać z tego przykładu macierz kosztu nie musi być symetryczna, gdyż np. koszt klasyfikacji klasy C_1 do C_2 wynosi 3 natomiast klasy C_2 do C_1 wynosi 2.

Niestety funkcja błędu E w postaci przedstawionej powyżej jest nieciągła, nie można więc zastosować metod opartych na minimalizacji gradientowej, do których należy wiele typów sieci neuronowych. Dlatego też często w zagadnieniach klasyfikacji stosuje się funkcje błędu wywodzące się z teorii aproksymacji. Typową funkcją stosowaną w tych metodach jest funkcja błędu średniokwadratowego

$$E(\mathbf{W}) = \frac{1}{2}\sum_i \sum_k (y_k(\mathbf{x}_i;\mathbf{W}) - d_i^k)^2 \tag{1.7}$$

gdzie d jest klasą zakodowaną za pomocą wartości numerycznych, sumowanie przebiega po wszystkich wektorach zbioru treningowego (indeks i), oraz po wszystkich wyjściach sieci (indeks k).

2.3. Generalizacja

Rozważmy granicę funkcji średniokwadratowej [5], w której wielkość zbioru treningowego N dąży do nieskończoności

$$\begin{aligned} E &= \lim_{N->\infty} \frac{1}{2N}\sum_{i=1}^{N}\sum_k (y_k(\mathbf{x}_i;\mathbf{W}) - d_i^k)^2 = \\ &= \frac{1}{2}\sum_k \int\int (y_k(\mathbf{x};\mathbf{W}) - d^k)^2 \, p(d^k,\mathbf{x}) d\mathbf{x} dd^k \end{aligned} \tag{1.8}$$

$p(d^k,\mathbf{x})$ jest gęstością prawdopodobieństwa w przestrzeni wejściowo-wyjściowej.

Korzystając z prawdopodobieństwa warunkowego możemy napisać

$$p(d^k,\mathbf{x}) = p(d^k \mid \mathbf{x})\, p(\mathbf{x}) \tag{1.9}$$

gdzie $p(d^k|\mathbf{x})$ oznacza gęstość prawdopodobieństwa dla danego d^k przy danej wartości wektora $\mathbf{x}$. Stąd

$$E = \frac{1}{2}\sum_k \int\int (y_k(\mathbf{x};\mathbf{W}) - d^k)^2 \, p(d^k \mid \mathbf{x})\, p(\mathbf{x}) d\mathbf{x} dd^k \tag{1.10}$$

Rozpisując powyższą różnicę kwadratów, a następnie wykonując poniższe podstawienia

$$< d^k \mid \mathbf{x} >= \int dp(d^k \mid \mathbf{x}) dd^k \tag{1.11}$$

$$< (d^k)^2 \mid \mathbf{x} >= \int (d^k)^2 \, p\left((d^k)^2 \mid \mathbf{x}\right) d(d^k)^2 \tag{1.12}$$

otrzymujemy

$$\begin{aligned} E = \frac{1}{2}\sum_k \int (y_k(\mathbf{x};\mathbf{W}) - < d^k \mid \mathbf{x} >)^2 \, p(\mathbf{x}) d\mathbf{x} + \\ + \frac{1}{2}\sum_k \int (< (d^k)^2 \mid \mathbf{x} > - < d^k \mid \mathbf{x} >^2) \, p(\mathbf{x}) d\mathbf{x} \end{aligned} \tag{1.13}$$

Jak widać drugi czynnik jest niezależny od stosowanego modelu, a więc oczywiście jest niezależny od parametrów adaptacyjnych. Reprezentuje on wariancję rozkładów pasujących do danych i wyznacza granicę najmniejszego możliwego błędu, jaki można osiągnąć.

Wartość pierwszego członu zależy od danych *D*. W rzeczywistości dane treningowe są skończone. Rozważmy grupę zbiorów treningowych, z których każdy składa się z *N* wektorów. Zbiory treningowe utworzone są poprzez próbkowanie tego samego rozkładu $p(\mathbf{x}|d)$. Dla prostoty rozważmy dane, których klasa zakodowana jest w postaci jednowymiarowego wektora *d*. Miarą dokładności dopasowania nauczonej sieci do danych jest pierwszy człon równania (1.13), jednak wartość ta zależy od odpowiednich danych *D*. Żeby wyeliminować tą zależność rozważymy średnią po wszystkich zbiorach danych

$$\mathrm{E}_D\left[(y(\mathbf{x}) - < d \mid \mathbf{x} >)^2 \right] \tag{1.14}$$

gdzie E_D oznacza wartość oczekiwaną w wyniku uśrednienia po wszystkich rozkładach danych *D*. Rozwińmy wyraz

$$\begin{aligned} (y(\mathbf{x}) - < d \mid \mathbf{x} >)^2 &= (y(\mathbf{x}) - \mathrm{E}_D[y(\mathbf{x})] + \mathrm{E}_D[y(\mathbf{x})] - < d \mid \mathbf{x} >)^2 = \\ &= (y(\mathbf{x}) - \mathrm{E}_D[y(\mathbf{x})])^2 + (\mathrm{E}_D[y(\mathbf{x})] - < d \mid \mathbf{x} >)^2 + \\ &\quad + 2(y(\mathbf{x}) - \mathrm{E}_D[y(\mathbf{x})])(\mathrm{E}_D[y(\mathbf{x})] - < d \mid \mathbf{x} >) \end{aligned} \tag{1.15}$$

Ponieważ rozpatrujemy wartość oczekiwaną powyższego wzoru to trzeci wyraz znika i otrzymujemy

$$\mathrm{E}_D[(y(\mathbf{x}) - < d \mid \mathbf{x} >)^2] = \left(\mathrm{E}_D[y(\mathbf{x})] - < d \mid \mathbf{x} >\right)^2 + \mathrm{E}_D\left[(y(\mathbf{x}) - \mathrm{E}_D[y(\mathbf{x})])^2 \right] \tag{1.16}$$

Pierwszy człon powyższego wzoru opisuje średnie (po wszystkich zbiorach danych) odchylenie modelu od wartości żądanej (nazywane w statystyce „obciążeniem modelu” [37]), natomiast drugi człon - wariancje modelu dla różnych danych *D*, a więc czułość modelu dla ustalonego zbioru danych.

Podczas poszukiwania parametrów adaptacyjnych modelu trzeba znaleźć optymalną granicę między dokładnością dopasowania do danych a złożonością (liczbą parametrów adaptacyjnych) modelu. Ograniczenie liczby parametrów adaptacyjnych prowadzi do tego, że funkcja, którą realizuje dany model staje się funkcją gładką. Model ma zbyt małe możliwości adaptacji, jest więc „obciążony” przyjętymi z góry założeniami co do formy funkcji. Z drugiej strony zbyt duża liczba parametrów powoduje zbytnie dopasowanie się modelu do danych i w konsekwencji dużą wariancję wyników w kolejnych próbach uczenia modelu oraz niestabilności modelu nawet przy drobnej zmianie danych treningowych.

Jedną z dróg zmniejszenia obydwu członów jest zwiększenie liczby danych. Powoduje to, że bez obawy możemy stosować bardziej złożone modele. Oczywiście przy założeniu, że liczba danych znacznie przewyższa złożoność modelu.

Ponieważ w realnych przypadkach nie mamy nieskończonego zbioru danych, to znalezienie optymalnej granicy jest trudne. Jedną z możliwości osiągnięcia tego celu jest narzucenie pewnych ograniczeń na model np. poprzez zastosowanie, o ile to jest możliwe, regularyzacji. Technika regularyzacyjna polega na dodaniu do minimalizowanej funkcji członu kary Ω. Całkowita funkcja błędu ma wówczas postać

$$E_C = E + \gamma\Omega \tag{1.17}$$

Postać członu regularyzacyjnego zależy od postaci funkcji $y(\mathbf{x})$. Wprowadzenie odpowiedniego członu Ω prowadzi do wygładzenia funkcji $y(\mathbf{x})$ np. wprowadzenie członu regularyzacyjnego postaci

$$\Omega = \frac{1}{2}\int\left(\frac{d^2y}{dx^2}\right)^2 dx \tag{1.18}$$

spowoduje znaczne wygładzenie funkcji y, ponieważ człon ten jest duży dla funkcji z dużą drugą pochodną, czyli dla funkcji mającej dużo oscylacji. Parametr γ reguluje wielkość wpływu czynnika kary na całkowitą funkcję błędu.

Alternatywną metodą kontroli złożoności modelu, w stosunku do regularyzacji, jest procedura wczesnego stopu. Podczas uczenia funkcja kosztu maleje. Ażeby uniknąć zbyt dokładnego dopasowania do danych, w trakcie uczenia model testowany jest na niezależnym zbiorze (zwanym zbiorem walidacyjnym). Jakość klasyfikacji na zbiorze walidacyjnym w początkowej fazie uczenia maleje wraz ze zmniejszającym się błędem na zbiorze treningowym. W pewnym momencie jednak, pomimo malejącego błędu na zbiorze treningowym, błąd na zbiorze walidacyjnym zaczyna rosnąć. Oznacza to, że system zbyt dokładnie dopasowuje się do danych i jego generalizacja maleje. Dlatego też w tym momencie należy przerwać proces uczenia. Używanie procedury wczesnego stopu, zakłada dostatecznie dużą złożoność modelu i początkową gładkość opisu. Pierwsze założenie można spełnić budując sieć o odpowiednio dużej liczbie parametrów adaptacyjnych, drugie natomiast (w przypadku sieci typu MLP) poprzez inicjalizowanej parametrów za pomocą małych wartości.

2.4. Metody oceny jakości klasyfikacji modeli uczących się.

Jak wynika z wcześniejszego rozdziału celem uczenie jest takie dopasowanie parametrów adaptacyjnych aby generalizacja była jak największa. Występuje wiele metod oceny generalizacji, a tym samym porównywania użyteczności algorytmów uczących. Metody te wybierane są w zależności od czasu uczenia, rozmiaru pliku uczącego oraz od tego czy występuje plik testowy. W przypadku użycia pliku testowego, po nauczeniu następuje sprawdzenie jakości klasyfikacji na tym pliku, dzięki czemu łatwo jest porównać kilka modeli klasyfikacyjnych.

Wszystkie sposoby oceny generalizacji w przypadkach, gdy brak jest oddzielnego zbioru testującego, oparte są na procesie losowego podziału danych na treningowe i testowe. Znane są trzy główne nieparametryczne metody statystycznej oceny generalizacji: kroswalidacja [21], kroswalidacja Monte Carlo, bootstrapping [52].

2.4.1. Kroswalidacja

Załóżmy, że mamy zbiór treningowy T złożony z rozłącznych podzbiorów S_i o zbliżonej liczebności, czyli $T = \left\{ \bigcup_i S_i ; \forall_{m \neq p} S_m \cap S_p = \varnothing, \forall_{m \neq p} |S_m| \approx |S_p| \right\}$ gdzie i=1...n. Wówczas n-krotna kroswalidacja polega na uczeniu klasyfikatora na zbiorze S_k^{TR} takim, że $S_k^{TR} = \left\{ \bigcup_{l \neq k} S_l \right\}$ i testowaniu na zbiorze $S_k^{TE} = \{S_k\}$ nie występującym w S_k^{TR}. Proces uczenia i testowania przebiega n-krotnie, czyli dla indeksu k=1...n. Końcowa jakość klasyfikacji wyliczana jest jako średnia z wartości otrzymanych dla każdego k. Wariancja wyników pozwala ocenić spodziewane odchylenia od średniej dokładności.

Szczególnym przypadkiem tej metody jest leave-one-out, w której każdy ze zbiorów S_i składa się z jednego elementu, jest to więc n-krotna kroswalidacja dla n równego liczbie wszystkich wektorów. Indeks k ma następujące wartości k=1...$|T|$ gdzie $|T|$ jest mocą zbioru treningowego. Metoda ta jest dużo bardziej kosztowna od kroswalidacji, dlatego też stosowana jest tylko dla małych zbiorów danych.

2.4.2. Kroswalidacja Monte Carlo

Jest to specjalny przypadek kroswalidacji, w którym część zbioru (najczęściej 2/3) w sposób losowy wybierana jest do zbioru treningowego a reszta do testowego. Po nauczeniu model jest testowany a jakość generalizacji zapamiętywana. Cały proces powtarzany jest dla różnych podziałów wielokrotnie (najczęściej 30 do 50 razy), a następnie końcowa wartość generalizacji jest wyznaczana jako średnia z wartości cząstkowych. W odróżnieniu od zwykłej kroswalidacji kolejne podziały nie opierają się na rozłącznych podzbiorach.

2.4.3. Bootstrapping

Jest to proces wybierania z podstawieniem. Jeżeli występuje m przypadków dokonywany jest w sposób losowy wybór m wektorów z ciągu treningowego z powtórzeniami, następnie po nauczeniu testuje się algorytm na pozostałej części, która nie wystąpiła w zbiorze treningowym. Podobnie jak kroswalidacja Monte Carlo kolejne podziały nie opierają się na rozłącznych podzbiorach.

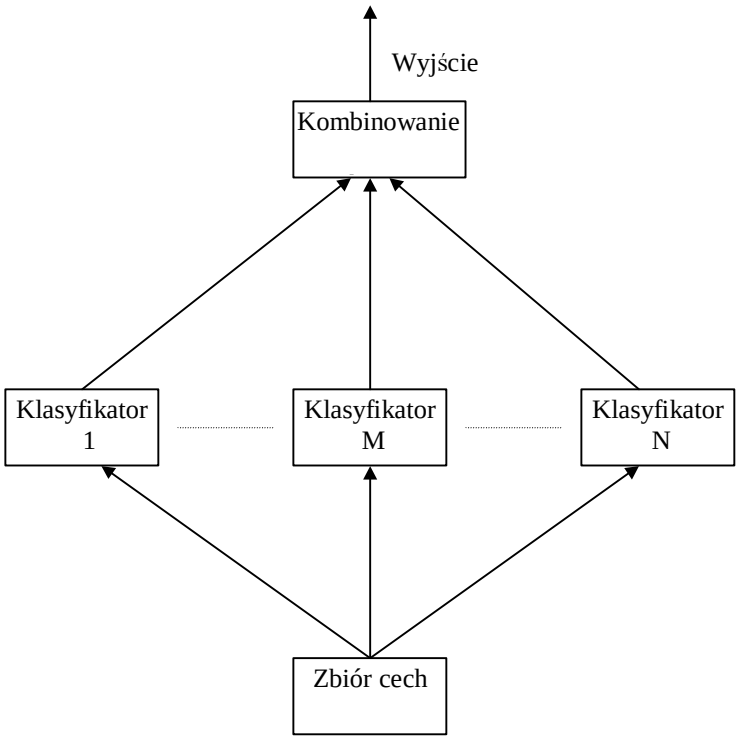
W przypadku niektórych z tych metod, z uwagi na losowy charakter podziału na plik uczący i testowy, otrzymuje się różne konfiguracje wektorów. Powoduje to, że za każdym razem w modelu powstaje inny zestaw parametrów adaptacyjnych. Dlatego też trzeba taką metodę powtórzyć kilkakrotnie, gdyż za każdym razem otrzymywany jest inny wynik. Końcowy wynik powstaje z uśrednienia wyników pośrednich.

W przypadku, gdy istnieje wydzielony plik testowy, ale uczenie polega na minimalizacji funkcji kosztu metodami gradientowymi, proces uczenia i testowania również trzeba powtórzyć kilkakrotnie. Jest to rezultatem wpadania danego modelu w lokalne minima, lub też moż-

liwości wystąpienia wielu równoważnych minimów globalnych, w wyniku czego otrzymuje się różne parametry klasyfikatora i różne wyniki na zbiorze treningowym i testowym.

2.5. Stabilność modeli klasyfikacyjnych

Niektóre modele klasyfikacyjne są niestabilne [7], co oznacza, że niewielkie perturbacje w ich zbiorach treningowych lub też w konstrukcji mogą powodować duże zmiany w konstruowanych obszarach decyzyjnych. Dokładność niestabilnych modeli może być poprawiona poprzez perturbacje zbioru danych treningowych lub poprzez kombinowanie modeli. W pierwszym przypadku tworzy się wiele wersji klasyfikatora na danych powstałych z perturbacji zbioru treningowego, a następnie kombinuje się te klasyfikatory w jeden. W drugim przypadku dokonuje się kombinacji różnych modeli klasyfikacyjnych. Kombinowanie klasyfikatorów realizowane jest w różny sposób: w literaturze statystycznej za pomocą metod takich jak stacking [69], bagging [9], arcing [8], w literaturze o sieciach neuronowych *commette networks* [46], w ekonometrii odpowiada temu metoda *forecast combining*, a w uczeniu maszynowym *evidence combination*.



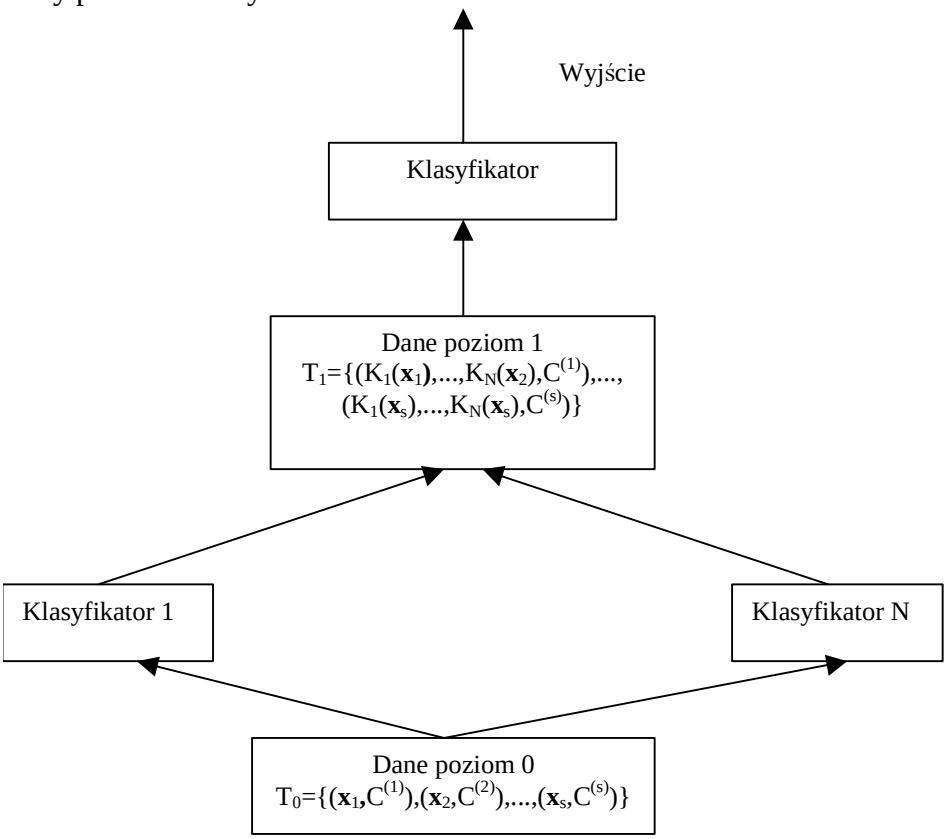
Rys 2:4 Schemat kombinowania klasyfikatorów

2.5.1. Stacking

Znane są dwie podstawowe formy tego algorytmu, jedna dotyczy pojedynczego klasyfikatora, a druga wielu klasyfikatorów. W obu formach występuje zbiór podziałów *p* zbioru treningowego *T*, $p_i = \{T_i^1, T_i^2\}, gdzie \bigvee_i T_i^1 \bigcup T_i^2 \subseteq T$

W algorytmie tym można użyć różnych technik dokonywania podziału zbioru treningowego, wybór techniki zależy od szybkości uczenia stosowanego modelu oraz od wielkości

zbioru treningowego *T*. Poniżej opisano algorytm stacking w wersji z kroswalidacją. Ogólny schemat tej metody przedstawia rysunek 2:5.



Rys 2:5 Schemat algorytmu stacking

Dane zbioru treningowego T_i^1 na poziomie 0 służą do uczenia zbioru klasyfikatorów na poziomie 0. Na podstawie wyjść tych klasyfikatorów dla zbioru testowego (powstałego przy kroswalidacji) T_i^2 tworzone są dane poziomu 1 i uczony jest klasyfikator poziomu 1. Cały proces tworzenia danych z poziomu 1 odbywa się w następujący sposób:

1. Dokonywana jest kroswalidacja zbioru T_0 na V części. Zbiór T_1 jest zbiorem pustym
2. Dla każdego etapu kroswalidacji *v* wykonywane są następujące kroki
   - Trenuj wszystkie klasyfikatory poziomu 0 na zbiorze treningowym $\boldsymbol{T_{0v}}$ (zawierającym *i* wektorów wejściowych).
   - Testuj wszystkie klasyfikatory na odpowiednim dla aktualnego zbioru treningowego zbiorze testowym (zawierającym *s* wektorów).
   - Dla każdego wektora z aktualnego zbioru testowego utwórz zbiór *S* postaci $S=\{K_1(x_1),...,K_N(x_1),C^{(1)},...,K_1(x_s),...,K_N(x_s),C^{(s)}\}$
   - $T_l \leftarrow T_l \bigcup S$

Użycie tylko wektorów testowych do tworzenia danych poziomu 1 umożliwia nauczenie klasyfikatora poziomu 1 błędów generalizacji powstałych na poziomie 0.

Nowy wektor najpierw wprowadzany jest na poziom 0. Wyjścia klasyfikatorów traktowane są jako dane dla poziomu 1, a wyjście klasyfikatora na poziomie 1 jako końcowa klasyfikacja.

2.5.2. Bagging

Jedną z najbardziej efektywnych metod perturbowania i kombinowania jest algorytm nazywany bagging [9]. Bagging dokonuje wielokrotnych perturbacji zbioru treningowego i generuje odpowiadające im klasyfikatory, a następnie kombinuje je za pomocą prostego głosowania.

Załóżmy, że dany jest zbiór treningowy *T* posiadający *N* elementów. Dla każdego przypadku ze zbioru treningowego ustalone zostaje prawdopodobieństwo $p = \frac{1}{N}$, przy użyciu, którego dokonujemy *N* krotnego próbkowania ciągu treningowego z podstawieniem (bootstrap), tworząc zbiór treningowy T_b. W wyniku tego próbkowania niektóre przypadki ze zbioru *T* mogą się nie pojawić w zbiorze T_b, niektóre natomiast mogą pojawić się wielokrotnie. Powtarzanie tej procedury prowadzi do powstania sekwencji niezależnych zbiorów treningowych, które służą do tworzenia różnych klasyfikatorów przy zastosowaniu tego samego algorytmu tworzącego klasyfikator.

Jak podaje L.Breiman w pracy [9], gdy klasyfikator jest dobry, pojedynczy klasyfikator może być daleki od optymalnego, natomiast kombinacje wielu dają klasyfikator bliski optymalnemu i stabilny. Niestety w przypadku słabych klasyfikatorów w wyniku kombinacji można otrzymać klasyfikator gorszy.

2.5.3. Arc-fs

Ulepszoną odmianą baggingu jest arc-fs [8], w którym startuje się tak jak poprzednio z prawdopodobieństwa $p = \frac{1}{N}$ i poprzez próbkowanie zbioru *T* tworzy zbiór *T1*. Następnie zwiększa się prawdopodobieństwo *p* przypadków, które są dotychczas błędnie klasyfikowane przez istniejące klasyfikatory, przez co zwiększy się ich liczba w zbiorze treningowym.

2.5.4. Komitet sieci

Przy zastosowaniu sieci neuronowych do analizy danych bardzo często trzeba wypróbować różne architektury sieci, jak również różne algorytmy uczenia. Na podstawie jakości generalizacji na zbiorze walidacyjnym wybierana jest najlepsza sieć, natomiast reszta sieci jest odrzucana. Takie postępowanie nie daje jednak pewności, że wybrana sieć, na nowym zbiorze testowym będzie miała najlepszą generalizacje, gdyż zbiór walidacyjny może być zaszumiony. Poza tym, poprzez odrzucenie pozostałych sieci duża część wysiłku związana z trenowaniem sieci neuronowej jest marnowana. Rozwiązaniem tych problemów jest zastosowanie tzw. komitetu sieci. W modelu tym można zastosować różne typy sieci, zarówno pod względem architektury czy też algorytmu uczenia, jak również dokładnie tą samą sieć, ale wytrenowaną do różnych minimów lokalnych.

Zaproponowanych zostało wiele różnych metod kombinowania klasyfikatorów np. ważone uśrednianie, kombinowanie na podstawie rzędów statystycznych, metoda głosowania itd. Wybór techniki kombinowania bardzo często zależy od interpretacji wyjścia klasyfikatorów np. metoda współczynników zaufania, występująca w literaturze uczenia maszynowego, nadaje się do klasyfikatorów, których wyjście może być traktowane jako miara zaufania do tego, że wzorzec należy do danej klasy. Istnieją również metody, które nadają się do dowolnego klasyfika-

tora np. metoda głosowania. W ogólności trudno jest odpowiedzieć na pytanie o ile wyżej wymienione metody poprawiają końcową klasyfikację. Jest to możliwe tylko dla pewnych szczególnych przypadków [66].

3. Wstępne przetwarzanie danych

W praktycznych zastosowaniach wstępne przygotowanie danych jest bardzo istotnym etapem w klasyfikacji danych mającym wpływ zarówno na szybkość uczenia modelu, jak również na jego późniejszą generalizację. Można wyróżnić trzy typy wstępnego przetwarzania danych: transformacje danych, uzupełnienie wartości brakujących i redukcja wymiarowości.

3.1. Transformacje danych

Analizowanie danych doświadczalnych związane jest z występowaniem różnego rodzaju skal pomiarowych, które mogą być symboliczne lub też numeryczne. Sieci neuronowe charakteryzują się tym, że wszystkie cechy opisujące analizowane obiekty muszą być numeryczne.

W przypadku modeli klasyfikacyjnych używających odległości jako miary podobieństwa bardzo często zdarza się, że poszczególne cechy charakteryzują jakiś stan fizyczny na podstawie różnych wielkości fizycznych, mających różne zakresy wartości, przez co mogą mieć one różny wpływ na odległość. Zastosować tu można kilka transformacji ujednolicających wpływ poszczególnych cech do wartości odległości. Do najbardziej znanych należy normalizacja oraz standaryzacja.

3.1.1. Normalizacja

Przeprowadzana jest zgodnie z poniższym wzorem

$$x'^{i} = \frac{x^i - x^i_{\min}}{x^i_{\max} - x^i_{\min}} \tag{2.1}$$

gdzie $x^i_{\max}$ jest maksymalną wartością występującą w zbiorze treningowym dla *i*-tej cechy, $x^i_{\min}$ minimalną wartością dla *i*-tej cechy. Normalizacja przeprowadzana jest dla wszystkich wektorów treningowym i testowych, przy czym dla obu zbiorów użyte są te same wartości $x^i_{\max}$, $x^i_{\min}$.

W wyniki normalizacji otrzymuje się wektory, których wartości cech są z zakresu [0,1]. Transformacja ta nie uwzględnia rozkładu wartości danej cechy, w związku z tym w przypadku wystąpienia kilku wektorów posiadających w tej cesze wartości znacznie różniące się od wartości typowych, w wyniku normalizacji następuje ściśnięcie większości wartości w tej cesze w bardzo wąskim przedziale.

3.1.2. Standaryzacja

Wykorzystanie rozkładu wartości w poszczególnych cechach prowadzi do transformacji zwanej standaryzacją.

$$x_i^{'} = \frac{x_i - \overline{x_i}}{\sigma_i(x)} \tag{2.2}$$

gdzie

$$\overline{x_i} = \frac{1}{n}\sum_j^n x_j^i \tag{2.3}$$

$$\sigma_i(x) = \frac{1}{n-1}\sum_j^n (x_i^j - \overline{x_i})^2 \tag{2.4}$$

W wyniku tej transformacji otrzymuje się cechy, których wartość średnia $\overline{x} = 0$, natomiast odchylenie standardowe σ=1, dzięki czemu wszystkie cechy mają jednakowy wkład do wartości odległości.

3.1.3. Obcinanie dokładności

W metodach bazujących na macierzy odległości np. metodzie *k*-NN czy metodzie dendrogramów, duża liczba wektorów treningowych znacznie spowalnia proces klasyfikacji i zwiększa wymagania na pamięć. W takich przypadkach można założyć, że część wektorów jest do siebie bardzo podobna tzn. odległość między nimi jest niewielka. Pozostawiając jeden wektor z grupy wektorów podobnych można dokonać zmniejszenia ich liczby w zbiorze treningowym. Wykrycie wektorów podobnych do siebie, bez kosztownego obliczania macierzy odległości, możliwe jest poprzez stopniowe obcinanie dokładności wszystkich cech (kwantyzacja poprzez obcinanie liczb [13]). Obiekty, które są bardzo podobne do siebie stają się identyczne, dzięki czemu można je odrzucić.

Obliczenia z reguły wykonywane są dla liczb zmiennoprzecinkowych mających 4 bajtową reprezentację, co dla danych przenormalizowanych do przedziału [0,1] daje rozdzielczość $r=10^{-7}$. Dla rozdzielczości *r* wszystkie dane z przedziału $[a,a+r]$ mają tą samą wartość.

Przed dokonaniem obcięcia dokładności wszystkie dane z ciągu treningowego przeskalowywane są do maksymalnego zakresu $[0,2^{32}]$

$$x = C(x_r - x_{\min});\ C = \frac{2^{32}}{x_{\max} - x_{\min}} \tag{2.5}$$

Aby uzyskać rozdzielczość $r = \frac{1}{2^k}$ musimy zostawić *k* znaczących bitów w x_r. Można to wykonać przez przesunięcie bitów

$$x(k) = x_{\min} + \frac{1}{C}\left(Round\left[\frac{x_r}{2^{33-k}} \right] \right) * 2^{33-k} \tag{2.6}$$

Dla k=1 wszystkie dane $0 \le x \le 1$ dla $x \le 0.5$ są mapowane na 0, a dla $x > 0.5$ na 1, natomiast dla k=32 otrzymujemy dane oryginalne. Dla k=2 dane w zakresie $x \in [0.25,0.75]$ zamieniane są na x_r=0.5, mniejsze na 0 a większa na 1, ogólnie otrzymujemy $1+2^{(k-1)}$ różnych wartości. Liczba znaczących bitów jest zwiększana tak długo, aż osiągnięta zostanie dopuszczalna liczba wektorów treningowych, lub też zostanie spełnione jakieś inne kryterium np. liczba skupisk powstała w wyniku zastosowania metody dendrogramów osiągnie swoją maksymalną wartość.

3.1.4. Rodzaje skal pomiarowych

Można wyróżnić cztery skale pomiarowe [24]:

- nominalna
- porządkowa
- interwałowa
- ilorazowa

W skali nominalnej poszczególnym wartościom nadawane są nazwy symboliczne. Dlatego też o dwóch obiektach można tylko powiedzieć, że w tej cesze są takie same lub też są różne np. płeć (mężczyzna, kobieta).

Skala porządkowa pozwala odróżnić dwa obiekty oraz stwierdzić znak tej różnicy, tj. mniejszy, jednakowy, większy. Jest ona liniowo uporządkowana. Podobnie jak w skali nominalnej nie jest znana odległość między nimi.

W skali interwałowej można dodatkowo ustalić różnice między wartościami cechy, czyli wykonać operacje dodawania i odejmowania. Na tej skali występuje zero umowne.

W skali ilorazowej dopuszczalne jest również mnożenie i dzielenie wartości skali. Punkt zerowy ma charakter bezwzględny.

W przypadku niektórych modeli klasyfikacyjnych istnieje konieczność zamiany jakieś skali pomiarowej na inną, gdyż dany model może np. pracować tylko z wartościami symbolicznymi lub tylko z wartościami numerycznymi.

Sieci neuronowe korzystają zawsze z wartości numerycznych cech. W przypadku występowania skali nominalnej lub porządkowej, nazwy symboliczne muszą zostać zakodowane na wartości numeryczne.

Skala nominalna

Najczęściej zmienną nominalną przedstawia się w postaci kolejnych liczb naturalnych np. kobieta=1, mężczyzna=2.

Inną metodą jest kodowanie wartości nominalnej w postaci wektora binarnego lub bipolarnego, którego długość równa jest liczbie możliwych atrybutów charakterystycznych dla danej cechy np. kodując cechę płeć otrzymuje się:

- kobieta 1 0 lub 1 -1
- mężczyzna 0 1 lub -1 1
- nie określona 0 0 lub –1 -1

Oczywiście w przypadku płci atrybut *nie określona* może nie mieć sensu, jednak w ogólności można wykorzystać tą postać zakodowanej wartości np. do zaznaczenia wartości brakującej, niezmierzonej. Niewątpliwą wadą tej metody, w przypadku sieci neuronowych, jest znaczne zwiększenie liczby cech wektorów treningowych, a co za tym idzie początkowej liczby parametrów adaptacyjnych.

Istnieje jeszcze jedna metoda, która zamienia wartości nominalne na prawdopodobieństwa występowania poszczególnych atrybutów, określone na podstawie liczby wystąpień w całym zbiorze treningowym. Odległość pomiędzy dwoma *n* wymiarowymi wektorami **A** i **B** dla *L* klasowego problemu wyliczana jest przy użyciu warunkowych prawdopodobieństw

$$D_v^{\alpha}(\mathbf{A},\mathbf{B}) = \sum_{j}^{n}\sum_{i}^{L} | \, p(C_i \mid A_j) - p(C_i \mid B_j) \, |^{\alpha} \tag{2.7}$$

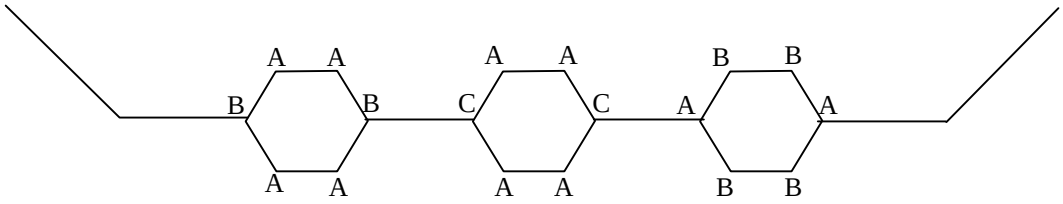
gdzie $p(C_i \mid A_j)$ wyznaczane jest na podstawie liczby wystąpień wartości A_j dla *j*-tej cechy występującej w wektorach należących do klasy C_i, podzielonej przez liczbę wystąpień wartości A_j dla wszystkich klas. Tak więc każda z wartości zamieniana jest na prawdopodobieństwa przynależności do poszczególnych klas.

Skala porządkowa

Skala porządkowa zazwyczaj kodowana jest w postaci kolejnych liczb naturalnych, oczywiście z uwzględnieniem odpowiedniej kolejności oryginalnych atrybutów np. mały, przeciętny, duży zostaje zakodowany na mały=1, przeciętny=2, duży=3.

3.2. Wartości brakujące

Przy analizie wielu danych doświadczalnych może pojawić się problem wartości brakujących tzn. w niektórych wektorach występują cechy, które z jakiegoś powodu nie zostały wyznaczone lub też nie mogły zostać wyznaczone z tego powodu, że dane cecha nie istnieje. Prostym przykładem drugiego przypadku są dane opisujące strukturę ciekłych kryształów [60]. Większość ciekłych kryształów można przedstawić za pomocą ogólnego szablonu postaci



Rys 3:1 Szablon reprezentujący strukturę ciekłych kryształów

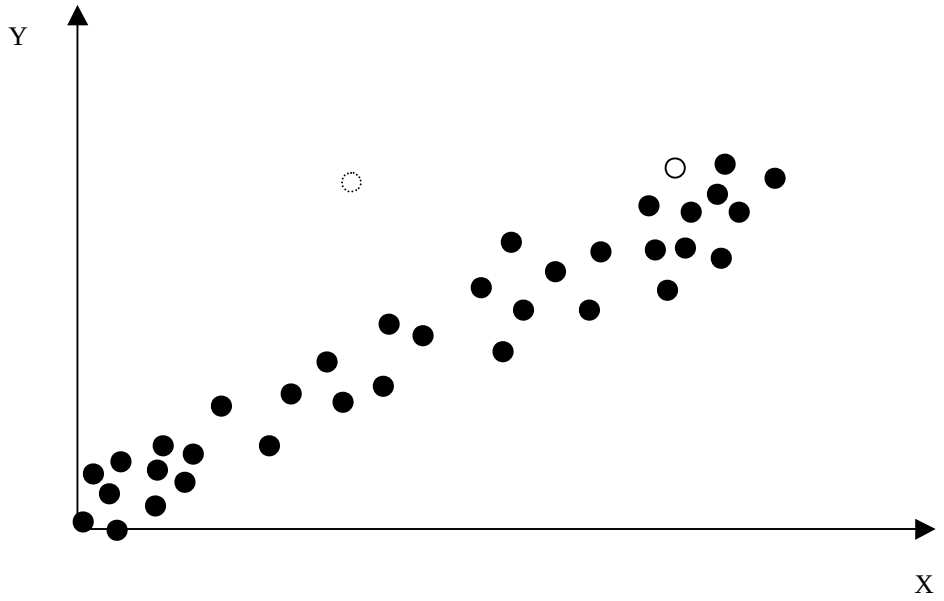
Występują tu 3 pierścienie i 2 mostki. W opisie ciekłych kryształów mamy cechy charakteryzujące poszczególne pierścienie i łączące je mostki. Bardzo wiele molekuł ciekłych kryształów nie posiada jednak wszystkich 3 pierścieni, a tylko 2 z nich. Dlatego też w miejscach opisujących pierścień nie występujący, pojawiają się wartości brakujące. Występowanie tego typu wartości brakujących zawiera informacje o strukturze, dlatego też nie są one przy analizie traktowane jako wartości brakujące. Zazwyczaj tego typu cechę koduje się jako wartość unikalną tzn. wartość, która nie jest nigdy osiągana przez żadną z cech.

Dużo bardziej interesujące są takie sytuacje, w których dane istnieją, lecz nie zostały zaobserwowane. Pomimo występowania takich sytuacji część klasyfikatorów może dokonać klasyfikacji. Radzą sobie z tym np. drzewa decyzji, istnieją jednak również takie jak np. sieci

neuronowe, które muszą mieć wszystkie cechy określone. Poza tym czasem może być interesujące dopełnienie wartości brakujących na podstawie istniejących obserwacji.

Najprostszą metodą pozwalającą użyć sieci neuronowe do danych z brakującymi wartościami, jest odrzucenia wszystkich wektorów zawierających braki. Oczywiście jest to możliwe tylko wtedy, gdy liczba takich wektorów nie jest duża w porównaniu z całym zbiorem treningowym.

Bardzo popularną metodą uzupełniania wartości brakujących jest wstawianie wartości uśrednionych dla odpowiedniej cechy, po wszystkich wektorach, które miały w tej cesze wartość określoną. Niestety jak pokazuje poniższy rysunek niebranie pod uwagę wartości pozostałych cech, prowadzić może do bardzo słabych rezultatów.



Rys 3:2 Uśrednianie wartości brakującej

Załóżmy, że mamy obiekt, którego położenie pokazane jest na rysunku za pomocą niewypełnionego kółka. Załóżmy teraz, że odrzucamy składową X tego obiektu tzn. uznajemy ją za wartość brakującą i na podstawie składowej Y oraz wszystkich pozostałych obiektów próbujemy wyznaczyć brakującą składową X. Jednak w wyniku uśrednienia po składowej X wszystkich wektorów otrzymamy położenie interesującego nas obiektu w miejscu wskazanym przez kółko narysowane przerywaną linią, a więc bardzo daleko od prawdziwego położenia.

Dużo lepszą estymację wartości brakującej można otrzymać przez wykorzystanie informacji znajdujących się w pozostałych cechach. Znanych jest kilka metod tego typu:

- Uzupełnienie wartości brakującej w oparciu o obiekt najbardziej podobny. Oczywiście określenie, które wektory są podobne zależy od używanej metody np. w przypadku algorytmu *k*-NN - jest to jakaś miara odległości. Metoda ta jest dość prosta, jednak poważną trudnością jest duża zależność wartości uzupełnianej od wybranej miary podobieństwa.
- Metoda regresyjna polega na zbudowaniu równania regresyjnego na podstawie przypadków posiadających wszystkie wartości dla danej zmiennej.
- Maksymalizacja wartości oczekiwanej EM (Expectation Maximization) zakłada postać prawdopodobieństwo łącznego dla X_{obs} i X_{brak}: $p(X_{obs},X_{brak}|\theta)$, gdzie X_{obs} oznacza wartości obserwowane, X_{brak} wartości brakujące istniejące w danych, θ parametry modelu. Stąd prawdopodobieństwo brzegowe dla X_{obs} wynosi

$$p(X_{obs} \mid \theta) = \sum_{X_{brak}} p(X_{obs}, X_{brak} \mid \theta) \tag{2.8}$$

Algorytm EM jest algorytmem iteracyjnym składającym się z dwóch kroków. W pierwszym kroku wyliczane jest prawdopodobieństwo dla wartości brakujących:

$$p_n(X_{brak}) = p(X_{brak} \mid X_{obs}, \theta^{n-1}) \tag{2.9}$$

gdzie θ^0, jest wyznaczone ze wzoru (2.10) przy założeniu że istnieją tylko wartości obserwowane.
W drugim kroku maksymalizuje się prawdopodobieństwo, tak jak gdyby nie było wartości brakujących i otrzymuje się nowe parametry modelu estymacji.

$$\theta^n = \max_{\theta} E\left(p(X_{brak,} X_{obs} \mid \theta)\right) \tag{2.10}$$

Kroki te powtarzane są tak długo, aż algorytm się zbiegnie tzn. zmiana parametrów modelu estymacji przy kolejnych iteracjach będzie zaniedbywalnie mała.

- Wielokrotne przypisywanie (MI multiple imputation) [54], [33]- jest to metoda podobna do EM, przy czym wprowadzony tu zostaje dodatkowo element niepewności. Każda wartość brakująca zastępowana jest przez zbiór m>1 możliwych do przyjęcia wartości otrzymanych z przewidywanego rozkładu. Metoda MI wymaga utworzenia m zbiorów, w których wartości obserwowane są wszędzie takie same natomiast wartości brakujące są wypełniane w różnych zbiorach różnymi liczbami. Różnice te odpowiadają niepewności, z którą każda z wartości brakujących może zostać przewidziana na podstawie wszystkich obserwowanych wartości.

3.3. Selekcja cech

Zazwyczaj informacje o obiekcie zbierane są w nadmiarze. W związku z tym występuje wiele zbędnych cech. Cechy mogą być zbędne, ponieważ nie wprowadzają żadnej nowej informacji lub też nie mają żadnego związku z celem klasyfikacji. Występowanie cech zbędnych powoduje niepotrzebne wydłużenie czasu uczenia oraz, w przypadku wielu modeli klasyfikacyjnych, prowadzi do trudności w znalezieniu optymalnego minimum funkcji kosztu, a co za tym idzie zmniejszenia generalizacji. Dlatego też w realnych zastosowaniach klasyfikacyjnych przed przystąpieniem do uczenia usiłuje się wykryć i usunąć tego typu cechy. Metody selekcji cech usiłują znaleźć minimalny podzbiór cech, które spełniają następujące kryteria:

1. jakość klasyfikacji nie ulegnie znaczącemu zmniejszeniu
2. rozkład klas otrzymany przy użyciu tylko wybranych cech jest tak bliski jak to tylko możliwe oryginalnemu rozkładowi tych klas.

Metody selekcji cech opisywane w literaturze można podzielić na metody formułowane na bazie:

1. statystyki i teorii informacji
   - teorii decyzji statystycznych
   - porównywania rozkładów prawdopodobieństwa
   - metody programowania matematycznego
   - teorii informacji
2. rozważań częściowo heurystycznych.

Przy założeniu statystycznego modelu problemu, jakość selekcji ocenia się najczęściej według kryterium ryzyka lub średniego prawdopodobieństwa błędu. Zastosowanie metod probabilistycznych do selekcji może być niemożliwe, jeśli odpowiednie rozkłady prawdopodobieństw nie są znane, co niestety w wielu realnych przypadkach ma miejsce.

Istnieje także możliwość użycia innych metod selekcji z pominięciem estymacji rozkładów. Wiele zagadnień selekcji można formułować na bazie łączenia selekcji z procesem szukania transformacji z przestrzeni pomiarów w inne przestrzenie, co wiąże się z tworzeniem nowych cech będących kombinacjami cech pierwotnych. Jest to spowodowane tym, że cechy, które umożliwiają dobre rozróżnienie obiektów z różnych klas są często trudne do wytypowania lub też do bezpośredniego pomiaru. Mogą natomiast być otrzymywane pośrednio za pomocą różnych transformacji, których rezultatem powinna być poprawa zawartości informacyjnej cech i niekiedy zmniejszenie liczby cech istotnych. Wadą tych metod jest brak wyraźnych związków między mierzalnymi cechami tzn. cechami pierwotnymi a klasyfikacją opartą na cechach otrzymanych po transformacji.

3.3.1. Selekcja cech w oparciu o teorie informacji

Jeżeli obserwowany obiekt opisany jest przez wektor **x** to entropia warunkowa klas ma postać:

$$H(C \mid \mathbf{x}) = \sum_{i=1}^{L} \left(-\log_2 p(C_i \mid \mathbf{x})\right) p(C_i \mid \mathbf{x}) \tag{2.11}$$

Zdefiniowana w ten sposób entropia warunkowa może być interpretowana jako miara nieoznaczoności klasy, do której należy obserwowany obiekt. Powyżej zdefiniowana entropia odnosi się tylko do jednego obserwowanego obiektu. Aby scharakteryzować miarę nieokreśloności klas, do których mogą należeć obserwowane obiekty, trzeba wziąć pod uwagę wszystkie możliwe obiekty **X**. Stąd wprowadza się średnią entropię warunkową postaci

$$H(\mathbf{C} \mid \mathbf{X}) = \sum_{x \in \mathbf{X}} H(\mathbf{C} \mid \mathbf{x}) p(\mathbf{x}) = \sum_{\mathbf{x} \in \mathbf{X}} \sum_{i=1}^{L} -\left(\log_2 p(C_i \mid \mathbf{x})\right) p(C_i, \mathbf{x}) \tag{2.12}$$

Występujące tutaj prawdopodobieństwo $p(C_i,\mathbf{x})$ jest prawdopodobieństwem łącznym występowania obiektów z klasy C_i opisanych za pomocą wektora **x**. Aby można było ocenić przydatność danych **X** do celów klasyfikacji konieczne jest porównanie średniej entropii warunkowej, gdy znane są **X**, z entropią, gdy dane te nie są znane. Ilość informacji o klasie, do której należy obserwowany obiekt, dostarczonej przez dane **X** ma postać

$$I(\mathbf{C}, \mathbf{X}) = H(\mathbf{C}) - H(\mathbf{C} \mid \mathbf{X}) \tag{2.13}$$

gdzie

$$H(\mathbf{C}) = \sum_{i=1}^{L} \left(-\log_2 p(C_i)\right) p(C_i) \tag{2.14}$$

gdzie $p(C_i)$ są to prawdopodobieństwa a priori poszczególnych klas.

Podaną wcześniej definicje informacji można również zastosować do oceny ilości informacji dostarczonej przez *i*-tą daną

$$I(\mathbf{C},\mathbf{x}_i) = H(\mathbf{C}) - H(\mathbf{C},\mathbf{x}_i) \tag{2.15}$$

Podzbiór cech, który maksymalizuje ilość informacji, zapewnia zwykle także najlepszą jakość klasyfikacji. Ze względu na określoną klasyfikację należy więc znaleźć taki podzbiór *k* cech, dla których ilość informacji osiąga maksimum. Rozwiązanie tego problemu wymaga w zasadzie obliczenia ilości informacji dla każdego spośród $\binom{n}{k}$ podzbiorów cech. Niestety liczba podzbiorów do przeszukania jest zazwyczaj bardzo duża np. niech *n=15, k=7*, to liczba podzbiorów do przeanalizowania wynosi 51480. Dlatego też stosuje się szereg uproszczeń pozwalających znacznie przyspieszyć proces selekcji.

Najprostszą metodą selekcji cech jest metoda „najpierw najlepszy". Polegająca na obliczaniu ilości informacji dostarczonej przez kolejne cechy z osobna. Spośród *n* cech wybiera się podzbiór *k* cech lokujących się na pierwszych *k* miejscach pod względem ilości informacji przez nie dostarczanych. Oczywiście wybrany tą drogą podzbiór cech nie musi być w ogólności optymalnym zbiorem cech. Algorytm ten nadaje się również do innych metod, które mogą określić jakąś funkcje użyteczności danej cechy.

Inną możliwością ograniczenia liczby analizowanych podzbiorów cech jest zastosowanie algorytmu sekwencyjnego. Konstrukcja poszukiwanego zbioru zaczyna się od wyszukania z całego zbioru cech, cechy dla której ilość informacji jest największa. W podobny sposób wybierana jest najlepsza para cech, przy czym para musi zawierać cechę wybraną w poprzednim etapie. W kolejnych etapach do wybranego zbioru cech dostawiane są kolejne, w taki sposób, aby zmaksymalizować ilość informacji dostarczanej przez wybrane cechy. Proces ten kontynuowany jest tak długo, aż wybrany podzbiór cech zawierać będzie ustaloną a priori liczbę cech.

Metodą odwrotną do podanej powyżej jest metoda kolejnej eliminacji cech. Kolejno odrzucane są cechy, które powodują najmniejszy spadek ilości informacji.

3.3.2. Selekcja cech w oparciu o rozwinięcie Karhunena-Loevego

Załóżmy, że w zbiorze treningowym istnieje *L* klas obiektów, których prawdopodobieństwa a priori są znane i wynoszą $p(C_i)$. Zakładamy, że każda z klas reprezentowana jest przez zbiór X_i (i=1,...,L), złożony z *n* wymiarowych wektorów **x** takich, że $\| \mathbf{x} \|^2 = 1$. Stąd można napisać

$$\underset{X_i}{E}\left[\| \underline{\mathbf{X}} \|^2 \right] = \sum_{j=1}^{n} \alpha_j^2(i) = 1; \;\; i = 1,...,L \tag{2.16}$$

gdzie

$$\alpha_j(i) = \underset{X_i}{E}\left[\underline{\mathbf{X}}\mathbf{e}_j \right] \tag{2.17}$$

$\mathbf{e}_j$ to ortonormalny wektor bazowy przestrzeni obserwacji, natomiast $\underline{\mathbf{X}} = (\underline{X}_1,...,\underline{X}_n)^T$ oznacza *n* wymiarową zmienną losową unormowaną, reprezentującą obserwowane obiekty. Własność sumowania do 1 współczynników liczbowych $\alpha_j^2(i)$ umożliwia traktowanie ich jako pewnego rodzaju wag wyrażających średni udział wektora bazowego $\mathbf{e}_j$ w tworzeniu danej obserwacji.

Wprowadźmy nowy układ współrzędnych posiadający wektory bazowe $\mathbf{v}_j$, $j = 1,...,n$. Niech układ ten będzie związany z układem wektorów $\mathbf{e}_j$ transformacją T_{jk} postaci

$$T_{jk} = \mathbf{e}_j^T \mathbf{v}_k \tag{2.18}$$

Jeżeli przez $\beta_j^2(i)$ oznaczymy odpowiednie współczynniki wagowe wektorów bazowych w nowym układzie współrzędnych, to uśredniona po wszystkich klasach waga współrzędnej $\mathbf{v}_j$ wynosi

$$\rho_j = \sum_{i=1}^{L} p(C_i)\beta_j^2(i) \tag{2.19}$$

Podobnie jak w przypadku współczynników $\alpha_j^2(i)$ wagi ρ_j spełniają warunek

$$\sum_{j=1}^{L} \rho_j = 1 \tag{2.20}$$

Współczynniki ρ_j mogą być interpretowane jako wskaźniki przydatności wektora bazowego v_j przy opisie klas w nowym układzie współrzędnych. Ze względu na poszukiwanie cech istotnych poszukujemy takiego układu współrzędnych, w którym część współczynników osiąga duże wartości, pozostałe zaś wartości małe, które mogą być wówczas pominięte. Chodzi, więc o osiągnięcie największego zróżnicowania współczynników ρ_j.

Z uwagi na własności współczynników ρ_j, można skonstruować wyrażenie $H(\mathbf{v})$ o strukturze identycznej do entropii zmiennej losowej

$$H(\mathbf{v}) = -\sum_{j}^{n} \rho_j \log \rho_j \tag{2.21}$$

W takim przypadku znalezienie współczynników ρ_j najbardziej zróżnicowanych sprowadza się do znalezienia minimum powyższej funkcji kryterialnej. Jak wykazał Watanabe, wektory bazowe optymalnego układu współrzędnych są wektorami własnymi macierzy kowariancji Q utworzonej na podstawie wszystkich obserwacji.

$$Q = \sum_{i}^{L} p(C_i) \underset{X_i}{E}\left[\underline{\mathbf{X}}\underline{\mathbf{X}}^T \right] \tag{2.22}$$

Jeżeli unormowane wektory własne macierzy Q zostaną uporządkowane według malejących odpowiadających im wartości własnych, to taki zbiór wektorów tworzy układ współrzędnych Karhunena-Loevego (K-L).

Występuje wiele metod selekcji cech opartych na rozwinięciu K-L, zwanymi również analizą czynników głównych (PCA- Principal Components Analysis). Wspólną własnością tych metod jest wyznaczenie nowego układu współrzędnych na bazie wektorów własnych macierzy kowariancji. Elementem odróżniającym poszczególne metody jest sposób konstrukcji tej macierzy na podstawie zbioru treningowego.

W chwili obecnej można wyróżnić kilka głównych typów metod selekcji cech, które korzystają z rozwinięcie K-L [58]. Jedną z najpopularniejszych jest *Selfic* metoda zaproponowana przez S. Watanabe. W metodzie tej macierz kowariancji konstruowana jest na podstawie obiektów **x** scentralizowanych ze względu na wszystkie klasy. Centralizowanie zbioru obserwacji dokonuje się przez odjęcie od każdego **x** wartości oczekiwanej postaci

$$\mu = \underset{X_0}{E}\left[\underline{\mathbf{X}}\right] \tag{2.23}$$

gdzie X_0 oznacza cały zbiór treningowy. Tak, więc macierz kowariancji oblicza się ze wzoru

$$\Sigma = \underset{X_0}{E}\left[(\underline{\mathbf{X}}-\mu)(\underline{\mathbf{X}}-\mu)^T\right] \tag{2.24}$$

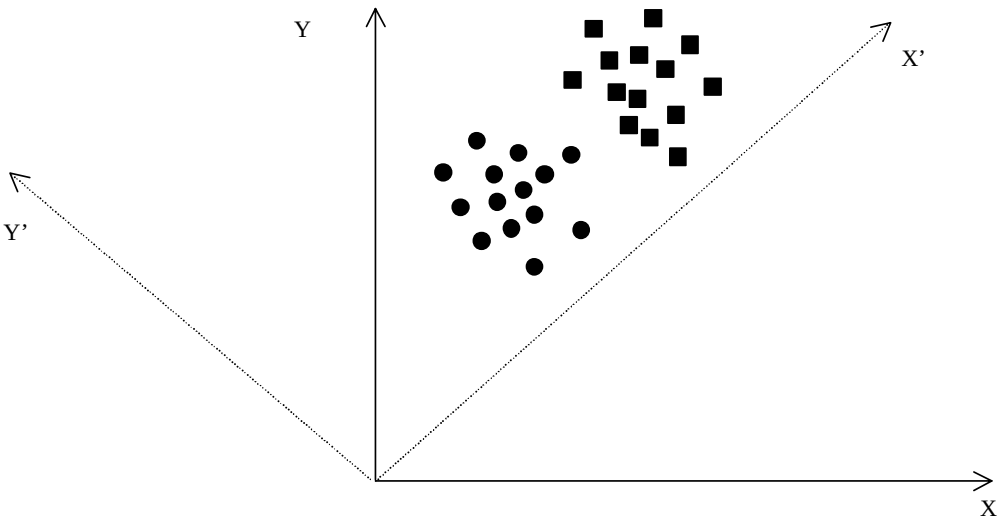
Inna metodę konstrukcji macierzy kowariancji zaproponowali Chien i Fu.

$$\Sigma = \sum_{i=1}^{L} p(C_i)\,\underset{X_i}{E}[(\underline{\mathbf{X}}-\mu_i)(\underline{\mathbf{X}}-\mu_i)^T] \tag{2.25}$$

gdzie μ_i określa wartości wektorów średnich klas.

Jeżeli wektory własne zostaną uporządkowane według malejących odpowiadających im wartości własnych to okazuje się, że ustalając liczbę cech $m<n$ uwzględnianych przy klasyfikacji, błąd średniokwadratowy aproksymacji i entropia osiągają minima dla m pierwszych cech w nowej uporządkowanej we wspomniany sposób przestrzeni.

PCA nadaje się do stosowania przede wszystkim w przypadku, gdy w danych występuje niewiele skupisk. Wówczas w wyniku działania PCA uzyskuje się układ współrzędnych, w którym łatwo odseparować poszczególne skupiska.



Rys 3:3 Ilustracja działania PCA

W przypadku, gdy występuje duża liczba skupisk PCA zawodzi i lepszymi metodami mogą okazać się metody dokonujące transformacji lokalnych.

3.3.3. Skalowanie wielowymiarowe (MDS)

Metoda wielowymiarowego skalowania (multidimensional scaling) [64], [70], [35] stosowana jest do rzutowania danych z przestrzeni wielowymiarowej *n* do przestrzeni niżej wymiarowej *q* ($n > q$).

Różnica podobieństwa pomiędzy parą obiektów (*i,j*) oznaczona jest zmienną δ_{ij}, która jest elementem macierzy (*NxN*) różnic podobieństwa Δ. Celem MDS jest rzutowanie danych w macierz konfiguracji **Y** o wymiarze (*Nxq*), przy czym wymiar przestrzeni cech *q* nie jest znany a priori.

Występują dwa główne modele MDS: niemetryczny i metryczny. W pierwszym z nich rzutowanie powinno być takie, że porządek w różnicach podobieństwa powinien odpowiadać porządkowi w odległościach. W drugim różnice podobieństwa w zbiorze rzutowanych punktów powinny być tak bliskie jak to tylko możliwe różnicom podobieństwa występującym w przestrzeni oryginalnej.

MDS niemetryczny

Warunek zachowania porządku różnic podobieństwa w przestrzeni konfiguracyjnej prowadzi do tego, że koniecznym jest tylko, aby funkcja różnicująca podobieństwa była funkcją odległości monotoniczno rosnącą. Zazwyczaj różnice podobieństwa δ_{ij} definiuje się w następujący sposób

$$\delta_{ij} = \left(\sum_k (x_{ik} - x_{jk})^2 \right)^{\frac{1}{2}} \tag{2.26}$$

W metodzie tej porządek w przestrzeni konfiguracji generowany jest poprzez minimalizacje pewnej szczególnej funkcji kosztu, lub miary STRESS-u, w sposób iteracyjny poprzez jakąś nieliniową procedurę optymalizacyjną.

Załóżmy, że mamy zbiór różnic podobieństwa danych δ_{ij}, i konfiguracje *N* punktów w *q* wymiarowej przestrzeni. Różnice podobieństwa mogą zostać uporządkowane stosownie do ich wielkości

$$\delta_{i_1 j_j} < \delta_{i_2 j_2} < \ldots < \delta_{i_M j_M}$$

gdzie $M=N(N-1)/2$.

Niech będą dane punkty w przestrzenie konfiguracyjnej **Y**, początkowo wygenerowane np. w sposób losowy, dla których policzona jest macierz odległości z elementami d_{ij}. Możliwe jest wyznaczenie zbioru różnic $\hat{d}_{ij}$, które są w przybliżeniu równe z d_{ij}, i wciąż zachowują własność monotoniczności odnośnie odpowiednich d_{ij}. Czyli

$$\hat{d}_{i_1 j_1} \le \hat{d}_{i_2 j_2} \le \ldots \le \hat{d}_{i_M j_M}$$

wielkości $\hat{d}_{ij}$ są monotonicznie zależne od d_{ij}, i dopasowanie tych wartości jest monotoniczną regresją .

W każdej fazie monotonicznej regresji różnice $\hat{d}_{ij}$ inicjalizowane są jako odległości d_{ij}, następnie porządkowane w taki sposób, że odpowiadające im różnice podobieństwa są w porządku rosnącym. Każda para wartości różnic porównywana jest wzajemnie i jeżeli występuje niepoprawny porządek są one kombinowane w grupę ze wspólną wartością różnicy równą średniej arytmetycznej kombinowanych wartości. Procedura kończy się, gdy nie ma już dalszego grupowania i różnice są równe lub odpowiadają narzuconemu porządkowi.

Kruskal zdefiniował funkcję bazującą na różnicach $\hat{d}_{ij}$ będącą miarą dopasowania odległości d_{ij} do $\hat{d}_{ij}$, postaci:

$$STRESS = \sqrt{\frac{\sum_{i<j}(d_{ij} - \hat{d}_{ij})^2}{\sum_{i<j} d_{ij}^2}} \tag{2.27}$$

Poprzez minimalizację funkcji *STRESS*-u znajdowane są współrzędne punktów w przestrzeni konfiguracyjnej.

Algorytm niemetrycznego MDS:

1. definiuj początkową konfigurację punktów $\mathbf{Y}(0)$, t=0
2. oblicz elementy macierz odległości d_{ij} dla aktualnej konfiguracji $\mathbf{Y}(t)$
3. wylicz elementy $\hat{d}_{ij}$
4. wylicz gradient funkcji *STRESS*, $\nabla STRESS$
5. wylicz nową konfigurację $\mathbf{Y}(t+1) = \mathbf{Y}(t) - \eta \nabla STRESS$
6. jeżeli algorytm zbiegł się to koniec w przeciwnym przypadku idź do punktu 2.

MDS metryczny

Najbardziej znanym przykładem metrycznej wersji algorytmu MDS jest algorytm mapowania Sammona. Algorytm Sammona pozwala dokonać mapowania z przestrzeni *n* wymiarowej w *q* wymiarową poprzez minimalizację funkcji postaci

$$E = \frac{1}{\sum_{i<j}^{N} D_{ij}} \sum_{i<j}^{N} \frac{(d_{ij} - D_{ij})^2}{D_{ij}} \tag{2.28}$$

gdzie D_{ij} reprezentuje elementy macierzy odległości w oryginalnej przestrzeni, natomiast d_{ij} elementy macierzy odległości w przestrzeni konfiguracyjnej.

4. Grupowanie

Grupowanie (klasteryzacja) jest zazwyczaj przedstawiane jako proces tworzenia skupisk (klastrów) elementów podobnych do siebie z obiektów fizycznych lub abstrakcyjnych klas. Do rozpoczęcia procesu grupowania potrzebna jest definicja miary podobieństwa pomiędzy obiektami, która zostanie użyta do wyznaczenia klas. Jako klasę w tym przypadku rozumie się zbiór obiektów, których podobieństwo wewnątrz klasy jest wysokie, natomiast między klasami niskie. Ponieważ kluczową rolę w takiej metodzie grupowania zajmuje definicja podobieństwa to metodę nazywa się metodą bazującą na podobieństwie, w odróżnieniu od metod określanych jako grupowanie koncepcyjne. Bardzo wiele metod bazujących na podobieństwie zostało rozwiniętych w taksonomii numerycznej.

Techniki grupowania same mogą zostać pogrupowane na wiele różnych sposobów. W jednym z podziałów bazujących na typie kontroli użytej podczas budowania skupisk wyróżnia się następujące kategorie: aglomeracyjną, podziałową i bezpośrednią.

Aglomeracyjna

Technika ta opiera się na postępującym zlewaniu się skupisk tzn. iteracyjnym łączeniu istniejących oddzielonych małych grup, by utworzyć coraz większe skupiska, aż pozostanie tylko jedno. Historia takiego łączenia przedstawiana jest zazwyczaj w postaci dendrogramu.

Poprzez zastosowanie jakiegoś progu określającego minimum podobieństwa proces łączenia może zostać przerwany zanim skupiska połączą się w jedno. Powstanie w ten sposób kilka skupisk, dla których wartość miary podobieństwa jest poniżej żądanego progu.

Podziałowa

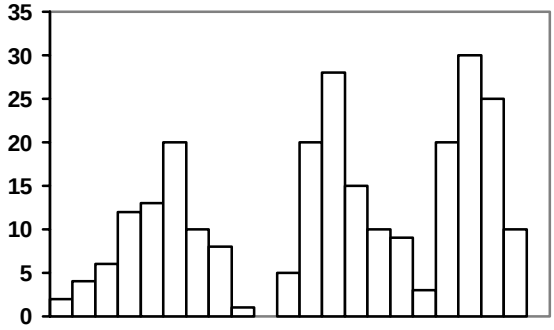
Tworzone są skupiska poprzez iteracyjny podział początkowego zbioru w coraz mniejsze, aż każdy element będzie tworzył pojedyncze skupisko.

Bezpośrednia

Technika bezpośrednia polega na znalezieniu z góry zadanej liczby skupisk. Trzeba przy tym znaleźć taki podział istniejących danych, aby był on optymalny z punktu widzenia miary określającej podział na skupiska. Jedną z pierwszych tego typu technik jest algorytm *k*-średnich MacQueena [38].

4.1. Metoda histogramowa

Jednym z algorytmów stosowanych w FSM (szczegółowy opis systemu znajduje się w rozdziale 8) do inicjalizacji sieci jest drzewo decyzyjne budowane na podstawie histogramu utworzonego dla każdego wymiaru osobno.



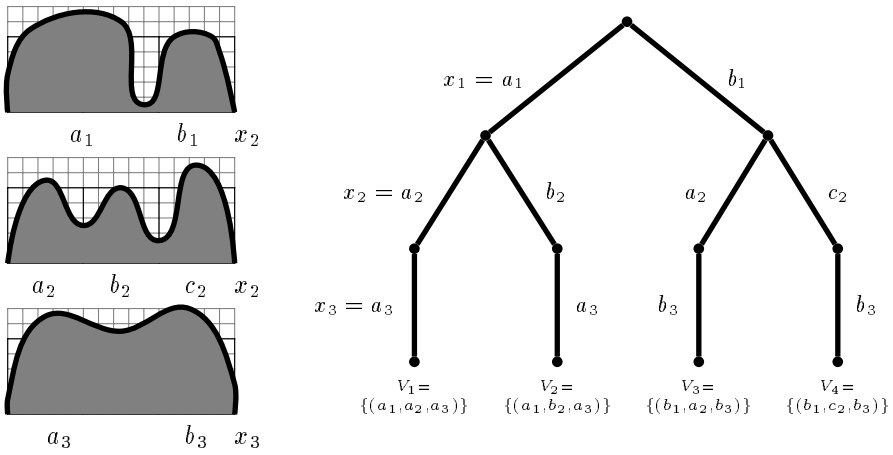
Rys 4:1 Histogram

Dany wymiar dzielony jest na *k* przedziałów (oś X na rysunku). W każdym przedziale zliczana jest liczba wektorów, która do niego wpada (oś Y na rysunku). Każdy z przedziałów może być łączony z sąsiednimi przedziałami, tworząc w ten sposób skupisko w przestrzeni jednowymiarowej. Jeżeli w danym wymiarze występuje wiele skupisk, to będą one oddzielone pustymi przedziałami lub też między kolejnymi przedziałami wystąpi wyraźna różnica w liczbie wpadających w nie wektorów (przedstawiono to na powyższym rysunku, gdzie można wyróżnić 3 skupiska). Załóżmy, że skupisko rozpoczyna się od *i*-tego przedziału i zajmuje *l* przedziałów, każdy o szerokości *s*. Wówczas pozycja oraz rozmiar skupiska w oryginalnej przestrzeni może zostać przedstawiona w następujący sposób

$$C_x = X_{\min} + s(i + \frac{1}{2}) \tag{3.1}$$

$$\sigma_x = \frac{1}{2} ls \tag{3.2}$$

$$s = \frac{| X_{\max} - X_{\min} |}{k} \tag{3.3}$$



Rys 4:2 Drzewo decyzyjne

Każde z tych jednowymiarowych skupisk może zostać następnie zrzutowane na wiele rozseparowanych *N*-wymiarowych skupisk. Podczas inicjalizacji wektory analizowane są w każdym wymiarze niezależnie, tak więc w pierwszym wymiarze dany wektor może należeć np. do skupiska C_1^i, w drugim wymiarze do skupiska C_2^j, aż wreszcie w *N*-tym wymiarze C_N^l. Każdy taki *N*-wymiarowy klaster może być przedstawiony jako łańcuch niżej wymiarowych skupisk $C_1^i -> C_2^j -> ... -> C_N^l$, stąd tego typu procedura tworzy drzewo decyzyjne. W liściach drzewa zliczane są wektory, które należą do danego *N*-wymiarowego skupiska. Gdy w przypadku któregokolwiek z wymiarów liczba klastrów jest zbyt duża, zmniejszona zostaje rozdzielczość *k*, co powoduje zmniejszenie liczby klastrów dla danego wymiaru. Gdy wszystkie klastry są już utworzone, obliczana jest odległość między nimi. Te skupiska, między którymi odległość jest mniejsza od zadanego progu łączone są w jedno.

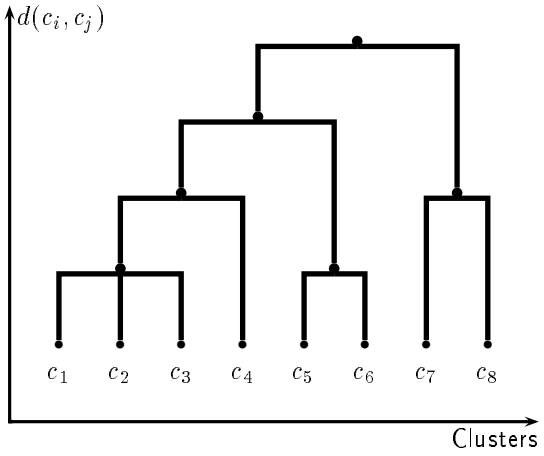
4.2. Dendrogram

W powyższym drzewie decyzyjnym rozdzielczość histogramów jest stopniowo zwiększana. Innym rozwiązaniem opartym na metodzie dendrogramów, które czasem prowadzi do lepszych rezultatów, jest stopniowe zmniejszanie rozdzielczości. Początkowo każdy wektor ciągu treningowego jest oddzielnym skupiskiem. Wyznaczona zostaje macierz odległości między poszczególnymi skupiskami, na jej podstawie najbliższe skupiska są ze sobą łączone. Znanych jest wiele metod wyznaczania odległości między skupiskami:

1. Na podstawie najbliższych obiektów należącymi do tych skupisk (*NN* nearst neighbor).
2. Na podstawie najdalszych obiektów należących do tych skupisk (*FN* furthest neighbor).
3. Obliczana jest jako średnia z odległości pomiędzy wszystkimi parami obiektów należących do analizowanych skupisk (*UPGMA* nweighted pair-group method using arithmetic).

4. Metoda podobna do *UPGMA* z tym, że wielkość skupisk (liczba elementów znajdujących się w skupisku) używana jest jako waga w obliczeniach (*WPGMA* weighted pair-group method using arithmetic averages).
5. Obliczana jest odległość między centrami skupisk. Centrum jest to uśrednione położenie wszystkich obiektów należących do tego skupiska (*UPGMC* unweighted pair-group method using the centroid average).
6. Podobnie jak w *UPGMC* z tym, że wielkość skupisk (liczba elementów znajdujących się w skupisku) używana jest jako waga w obliczeniach (*WPGMC* weighted pair-group method using the centroid average)

Grupowanie polegające na tym, że na jednym poziomie łączy się istniejące skupiska i zastępuje je przez skupiska na wyższych poziomach, nazywane jest grupowaniem hierarchicznym. Cała procedura trwa tak długo, aż liczba skupisk jest wystarczająco mała, lub też aż odległość między skupiskami jest większa od zadanej progowej. Poniższy rysunek przedstawia schematycznie ideę dendrogramów.



Rys 4:3 Dendrogram

Poziom pierwszy reprezentuje stan początkowy, w którym każdy z dziesięciu elementów tworzy osobne skupisko. Za wyjątkiem węzłów początkowych (na pierwszym poziomie) każdy z istniejących węzłów reprezentuje operację łączenia węzłów z niższego poziomu. Każdemu węzłowi można więc przypisać wartość numeryczną (np. odległość znajdującą się na osi Y, lub poziom).

4.3. K średnich klasterów

Załóżmy, że mamy N punktów i chcemy znaleźć k reprezentatywnych wektorów μ_j j=1,...,k. Algorytm poszukuje takiego podziału zbioru wektorów **x** w k rozłączne podzbiory S_j zawierające N_j elementów, dla których poniższa funkcja osiąga minimum

$$J = \sum_{j=1}^{k} \sum_{l \in S_j} \| \mathbf{x}^l - \mu_j \|^2 \tag{3.4}$$

gdzie μ_j jest średnią wektorów znajdujących się w podzbiorze S_j

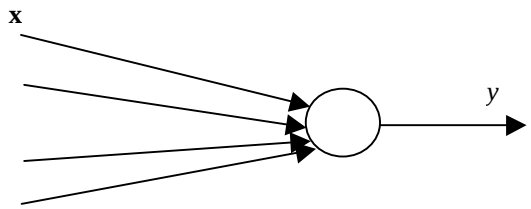
$$\mu_j = \frac{1}{N_j} \sum_{l \in S_j} \mathbf{x}^l \tag{3.5}$$

Algorytm rozpoczyna od losowego podziału punktów na *k* zbiorów, a następnie obliczane są wektory średnie. Wszystkie punkty przydzielane są ponownie do zbiorów S_j. Dany wektor *l*-ty przydzielony zostaje do tego zbioru, do którego odległość między nim a wektorem średnim tego zbioru jest najmniejsza. Gdy wszystkie punkty zostaną sprawdzone, ponownie oblicza się średnie zbiorów S_j. Cała procedura powtarzana jest tak długo, aż żaden z wektorów nie zmieni swej przynależności.

5. Propagacja wsteczna

5.1. Neurony

Podstawowym elementem każdej sieci neuronowej są neurony, czyli elementy przetwarzające sygnały, nazywane również węzłami sieci. Neurony łączone są ze sobą tworząc bardziej skomplikowane struktury zwane siecią neuronową. Każdy neuron ma wiele wejść i jedno wyjście.



Rys 5:1 Model neuronu

Sygnał wyjściowy *y* wypływający z węzła jest funkcją sygnałów wejściowych **x** wpływających do węzła oraz wag, które z danym węzłem są związane.

$$y = f(\mathbf{x}, \mathbf{w}) \tag{4.1}$$

W najprostszym przypadku może to być funkcja liniowa postaci

$$y = \sum_i w_i x_i \tag{4.2}$$

Funkcja ta nazywana jest funkcją transferu i ma na ogół charakter nieliniowy. W zależności od konkretnego celu, jakiemu służy neuron, może przyjmować różne postacie.

Pierwszy model sztucznego neuronu, który inspirowany był przez model biologiczny, opisany został przez McCulloch-a i Pitts-a. Zaproponowany przez nich neuron posiadał progową funkcję transferu postaci

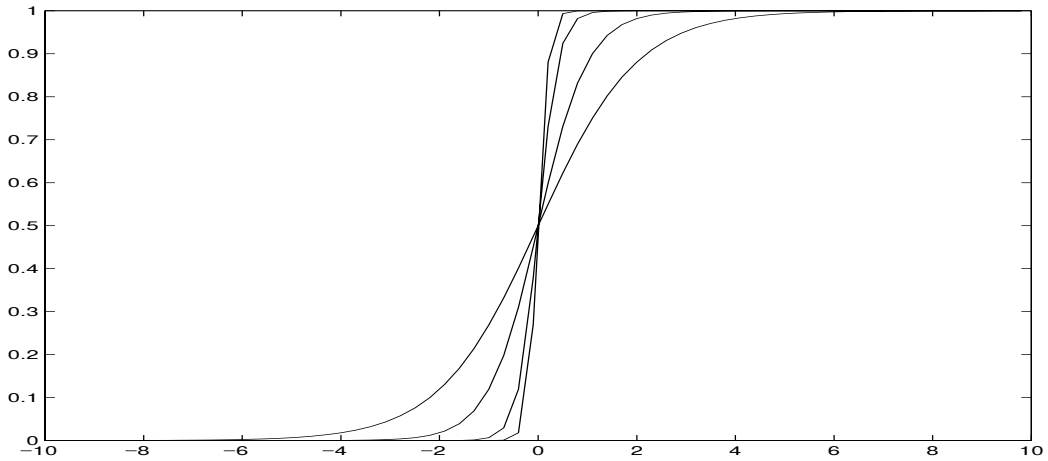
$$y = \begin{cases} 1 \quad gdy \quad \mathbf{wx} + \theta \geq 0 \\ 0 \quad gdy \quad \mathbf{wx} + \theta < 0 \end{cases} \tag{4.3}$$

gdzie θ jest progiem wzbudzenia. Zaletą tej funkcji jest to, że działanie sieci sprowadza się do realizacji złożonych funkcji logicznych. Wadą natomiast jest brak pochodnej, co uniemożliwia stosowanie tej funkcji w sieciach, w których stosuje się metody gradientowe.

Najczęściej stosowaną funkcją transferu w sieciach propagacji wstecznej jest nieliniowa funkcja sigmoidalna zwana również logistyczną

$$y = \sigma(\mathbf{wx} + \theta) = \frac{1}{1 + e^{-s(\sum_i w_i x_i + \theta)}} \tag{4.4}$$

gdzie *s* to skos funkcji sigmoidalnej, θ próg wzbudzenia, $\sum_i w_i x_i$ sygnał wejściowy.



Rys 5:2 Wykres funkcji sigmoidalnej dla skosów o wartościach 1,2,5,20

Powyższy rysunek przedstawia wykres funkcji sigmoidalnej dla różnych skosów. Wraz ze wzrostem skosu następuje wyostrzenie funkcji sigmoidalnej, a przy bardzo dużych skosach następuje przejście do funkcji skokowej.

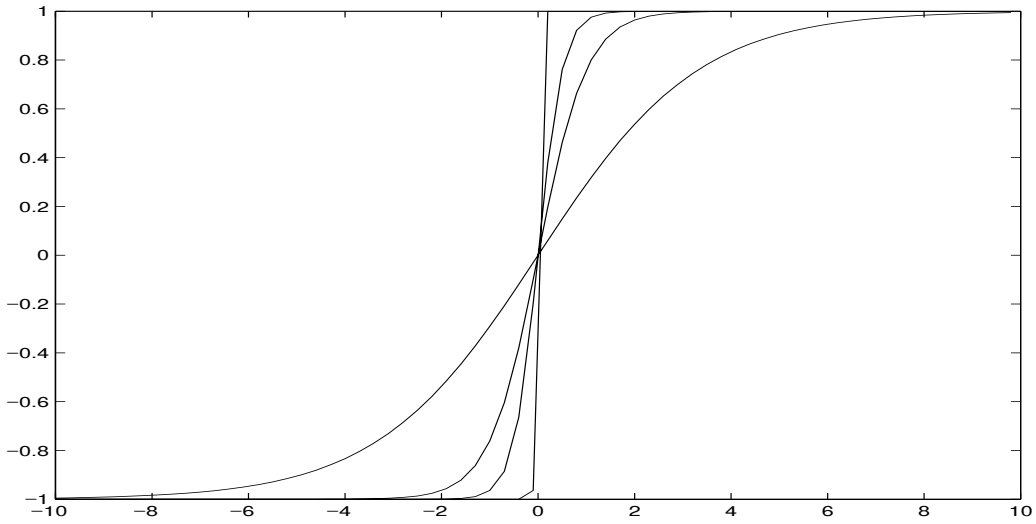
Na ogół w algorytmie MLP w trakcie uczenia optymalizowane są tylko wagi oraz próg, natomiast skos ma wartość ustaloną na 1. Mimo to istnieje możliwość manipulacją skosu bez zmiany skosu explicite. Efektywny skos zależny jest od normy wag i wielkości progu:

$$s(\mathbf{wx}+\theta) = \mathbf{w}'\mathbf{x}+\theta' \tag{4.5}$$

Informacja o skosie może więc być zawarta w wagach oraz progu. W przypadku, gdy wystarczająca jest funkcja gładka, wagi oraz próg będą odpowiednio małe. Natomiast, gdy np. dwie klasy są bardzo blisko siebie i do rozseparowania ich potrzebna jest ostra sigmoida, wagi oraz próg będą duże.

Inną bardzo popularną funkcją transferu stosowaną w sieciach MLP jest funkcja postaci

$$y = \tanh\left(s(\sum_i w_i x_i + \theta)\right) = \frac{e^{s(\sum_i w_i x_i+\theta)} - e^{-s(\sum_i w_i x_i+\theta)}}{e^{s(\sum_i w_i x_i+\theta)} + e^{-s(\sum_i w_i x_i+\theta)}} \tag{4.6}$$

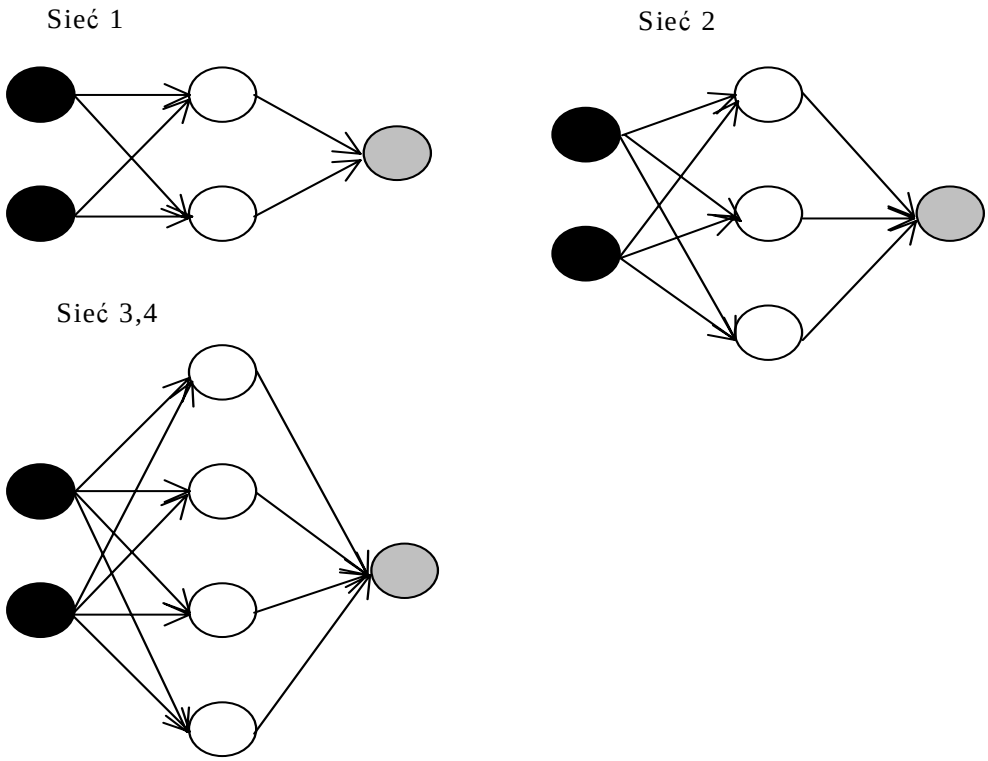


Rys 5:3 Wykres funkcji tanh dla skosów o wartościach: 0.3,1,2,20

5.2. Architektura sieci

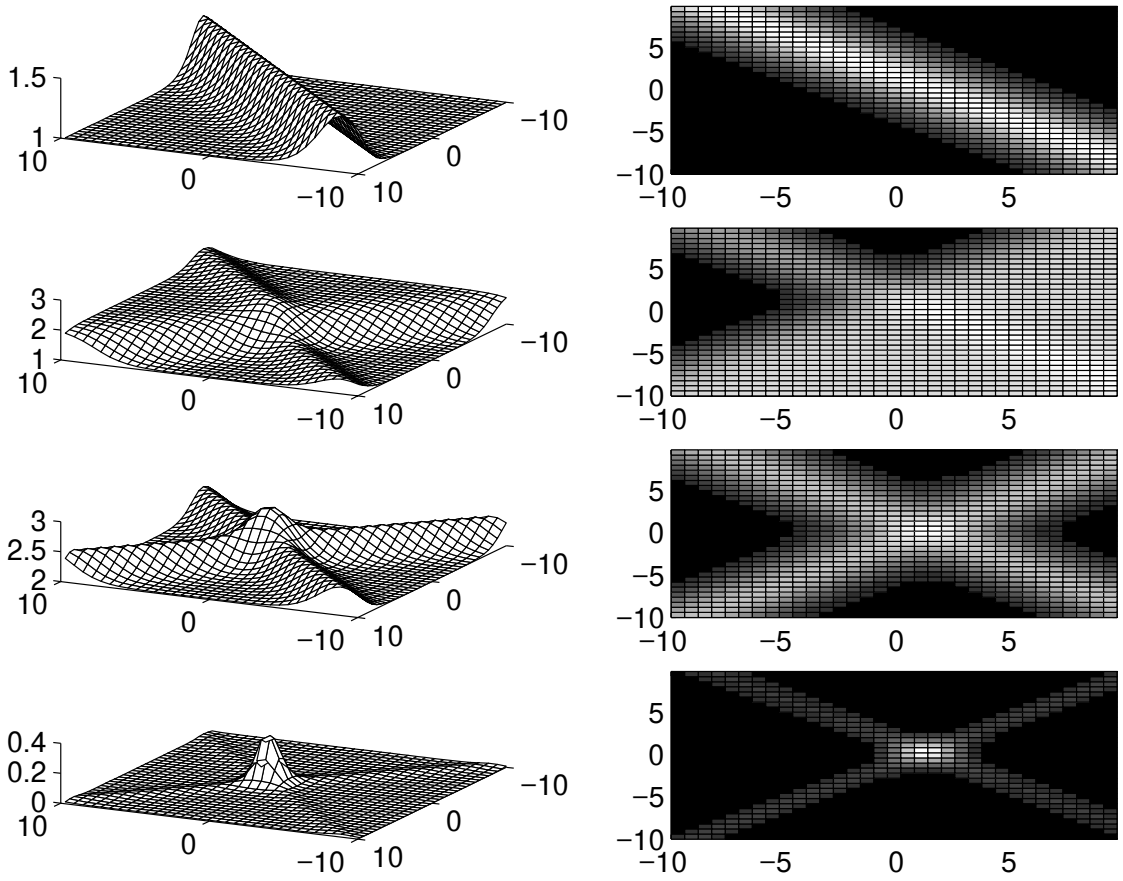
Neurony łączone są między sobą tworząc w ten sposób sieć. Liczba i sposób połączenia neuronów określa architekturę sieci neuronowej. Węzły występujące w sieciach MLP tworzą warstwy. Wyróżnia się trzy typy warstw: wejściowe, ukryte i wyjściowe. W warstwie wejściowej występują neurony, które nie wykonują żadnych obliczeń i do których bezpośrednio wprowadzane są dane treningowe. Z uwagi na brak obliczeń w opisie sieci warstwa ta jest pomijana. W warstwie ukrytej występują węzły, które nigdy nie są połączone bezpośrednio z wektorem treningowym oraz z wyjściem sieci. W warstwie wyjściowej natomiast występują węzły mające bezpośrednie połączenie z wyjściem sieci.

Łączenie węzłów w sieć umożliwia, w zależności od sposobu połączenia i liczby węzłów w sieci, tworzenie różnych obszarów decyzyjnych. Poniżej przedstawione są 4 przykłady prostych sieci i wykresy realizowanych przez nie funkcji



Rys 5:4 Przykład kilku architektur sieci MLP

W sieci pierwszej występują dwa neurony ukryte realizujące funkcje sigmoidalne, węzeł wyjściowy jest liniową kombinacją węzłów ukrytych, sieć druga zawiera 3 węzły ukryte i węzeł wyjściowy dokonujący liniowej kombinacji. Sieć trzecia działa podobnie jak dwie poprzednie tylko występują cztery węzły ukryte i wreszcie sieć czwarta jest dokładnie taka sama jak trzecia z tą zmianą, że węzeł wyjściowy realizuje funkcję sigmoidalną o dużym skosie.



Rys 5:5 Obszary decyzyjne

W pierwszych trzech przypadkach funkcje realizowane przez sieć są zdelokalizowane, w czwartym przypadku natomiast mamy funkcję zlokalizowaną, której niezerowe wartości występują w bardzo niewielkim obszarze przestrzeni wejściowej. Widać stąd, że sposób łączenia węzłów w sieci, funkcje realizowane przez te węzły, oraz liczba węzłów ma istotny wpływ na kształt obszarów decyzyjnych realizowanych przez sieć. Stąd też mogą pojawić się trudności związane z realizacją pewnych kształtów decyzyjnych przez sieć MLP. Każdy obszar decyzyjny budowany jest przez kombinacje różnych funkcji realizowanych przez neurony. W przypadku użycia funkcji sigmoidalnej do utworzenia obszaru decyzyjnego o kształcie okręgu w dwóch wymiarach, potrzeba co najmniej trzech neuronów, im dokładniejszy okręg będziemy chcieli zbudować, tym więcej neuronów będzie potrzeba. Istnieje możliwość zmiany kształtów obszarów decyzyjnych realizowanych przez jednostki sigmoidalne poprzez transformacje wykonaną na danych treningowych. Opis tego typu sieci znajduje się w późniejszym rozdziale pt. sieci nieeuklidesowe.

Jak widać z poprzednich rozważań przed rozpoczęciem uczenia trzeba zbudować sieć. W tym celu należy ustalić liczbę warstw ukrytych, liczbę węzłów w poszczególnych warstwach ukrytych oraz sposób ich połączeń.

Wyznaczenie liczby węzłów w warstwie wyjściowej jest proste. W przypadku klasyfikacji najlepszym rozwiązaniem jest zakodowanie poszczególnych klas za pomocą binarnego wektora o długości równej liczbie klas np. w przypadku wystąpienia 3 klas, klasa 1 kodowana jest jako 1 0 0, druga 0 1 0, trzecia 0 0 1. Określenie liczby węzłów ukrytych jest znacznie trudniejsze i na ogół wymaga wypróbowania różnych architektur.

Istnieje oszacowanie [28] pozwalające wyznaczyć dla sieci MLP o ustalonej architekturze (ustalonej liczbie parametrów adaptacyjnych) wielkość zbioru treningowego zapewniające-

go dobrą generalizację. Pojęcie dobra generalizacja oznacza generalizację bliską najlepszej możliwej przy zadanej złożoności sieci. Oszacowanie to mówi, że sieć będzie miała prawie na pewno dobrą generalizację pod warunkiem, że spełnione będą następujące założenia:

1. błąd na zbiorze treningowym jest mniejszy niż $\frac{\varepsilon}{2}$
2. liczba przypadków w ciągu treningowym *N* spełnia następującą relacje

$$N \geq \frac{32w}{\varepsilon} \ln\left(\frac{32H}{\varepsilon}\right) \tag{4.7}$$

gdzie *w* to liczba parametrów adaptacyjnych w sieci, ln oznacza logarytm naturalny, *H* całkowita liczba węzłów ukrytych.

Relacja ta pozwala oszacować liczbę przypadków w ciągu treningowym potrzebną do osiągnięcia dobrej generalizacji. W rzeczywistości natomiast liczba przypadków może być znacznie mniejsza. Dlatego też zbudowanie dobrej sieci dla ustalonych danych, wymaga wielu prób z różnymi architekturami, co jest procesem czasochłonnym. Jest to bardzo poważna wada tego typu sieci. Aby osiągnąć dużą zdolność do generalizacji, trzeba wbudować w sieć jak najwięcej wiedzy o problemie i ograniczyć liczbę połączeń.

W rzeczywistości koncentrujemy się głównie na zastosowaniu najmniejszej możliwej liczby neuronów (zmniejszeniu złożoności sieci), co zredukuje czas obliczeń, jak również prawdopodobnie poprawi generalizację. Jednym z rozwiązań jest wyszukanie optymalnej architektury w przestrzeni możliwych architektur. Robi się to np. poprzez zastosowanie algorytmów genetycznych. Jednak każdą badaną architekturę trzeba nauczyć i oszacować dla niej funkcję kosztu, czyli jakość jej działania i liczbę parametrów adaptacyjnych. Jest to proces bardzo kosztowny i raczej rzadko używany w realnych zastosowaniach.

Mniej czasochłonnym jest proces zmiany architektury sieci w trakcie uczenia. Występują dwie techniki takiego postępowania: konstruowanie sieci poprzez dodawanie jednostek w trakcie uczenie (tzw. algorytmy konstruktywistyczne) lub też start z sieci mającej dużą liczbę wag a następnie:

- poprzez konstrukcję odpowiedniej funkcji kosztu zmniejszanie liczby parametrów adaptacyjnych.
- po zakończonym uczeniu następuje likwidowanie połączeń, których usunięcie nie powoduje znacznego wzrostu błędu np. OBD (Optimal Brain Damage), OBS (Optimal Brain Surgeon).

5.3. Funkcje błędu

W algorytmie propagacji wstecznej uczenie polega na minimalizacji odpowiedniej funkcji kosztu. Występuje wiele typów tego rodzaju funkcji, wybór jakieś funkcji jest zależny od celu, który chcemy osiągnąć. Najprostsza definicja funkcji celu ma postać błędu średniokwadratowego, która przy założeniu tylko jednego wyjścia sieci ma postać

$$E = \frac{1}{2}\sum_i (y(\mathbf{x}_i, \mathbf{w}) - d_i)^2 \tag{4.8}$$

W przypadku natomiast, gdy zadaniem układu jest minimalizacja największego odchylenia odpowiedzi od wielkości żądanej, stosuje się wyższe potęgi błędu w definicji funkcji kosztu

$$E = \sum_i (y(\mathbf{x}_i, \mathbf{w}) - d_i)^{2m} \tag{4.9}$$

Jednym z potencjalnych niebezpieczeństw mogących pojawić się przy wartości $m \geq 1$ jest to, że dominujący wpływ mają dane, dla których błąd jest największy. To oznacza, że punkty mające szczególnie duże błędy (znane w statystyce jako „outliers") doprowadzą do błędnego rozwiązania. Jednym z możliwych sposobów powodujących zmniejszenie czułości na takie punkty jest obniżenie potęgi wykładnika. Przykładem takiej funkcji jest funkcja wykorzystująca normę L_1, umożliwiająca bardziej równomierny udział poszczególnych składników błędu

$$E = \sum_i | y(\mathbf{x}_i, \mathbf{w}) - d_i | \tag{4.10}$$

Założymy, dla uproszczenia, że występują dwie klasy i rozpatrujemy sieć z dwoma wyjściami, pierwsze wyjście d reprezentuje prawdopodobieństwo a posteriori $p(C_1|x)$ przynależności wektora x do klasy C_1. Drugie wyjście reprezentuje prawdopodobieństwo drugiej klasy ma oczywiście postać $p(C_2|x)=1-d$. Wówczas funkcja kosztu, która w porównaniu do funkcji średniokwadratowej lepiej nadaje się do problemu klasyfikacji ma postać funkcji entropii

$$E = \sum_i \left[d_i \ln y(\mathbf{x}_i, \mathbf{w}) + (1 - d_i) \ln(1 - y(\mathbf{x}_i, \mathbf{w})) \right] \tag{4.11}$$

ponieważ minimum funkcji jest dla $y(\mathbf{x}_i,\mathbf{w})=d_i$ czyli

$$E_{\min} = -\sum_i \left[d_i \ln d_i + (1 - d_i) \ln(1 - d_i) \right] \tag{4.12}$$

to w celu przesunięcia minimum funkcji do zera stosuje się funkcję postaci

$$E = -\sum_i \left[d_i \ln \frac{y(\mathbf{x}_i, \mathbf{w})}{d_i} + (1 - d_i) \ln \frac{1 - y(\mathbf{x}_i, \mathbf{w})}{1 - d_i} \right] \tag{4.13}$$

Zapiszmy błąd sieci dla danego i-tego wzorca w postaci

$$y(\mathbf{x}_i, \mathbf{w}) = d_i + \varepsilon_i \tag{4.14}$$

co prowadzi do

$$E = -\sum_i \left[d_i \ln \left(1 + \frac{\varepsilon_i}{d_i} \right) + (1 - d_i) \ln \left(1 - \frac{\varepsilon_i}{1 - d_i} \right) \right] \tag{4.15}$$

Ponieważ rozpatrujemy klasyfikacje, w której klasy reprezentowane są przez wektory binarne, czyli *d* może przyjmować wartości 1 i 0 to

$$E = -\sum_{i \in C_1} \ln(1+\varepsilon_i) - \sum_{i \in C_2} \ln(1-\varepsilon_i) \tag{4.16}$$

Przy założeniu, że błąd jest mały funkcja kosztu ma postać

$$E \approx \sum_i |\varepsilon_i| \tag{4.17}$$

Minimalizacja entropii powoduje, że wszystkie błędy są traktowane z tą samą wagą w przeciwieństwie do funkcji będącej sumą kwadratów czy też wyższych potęg, gdzie duże błędy są bardziej istotne.

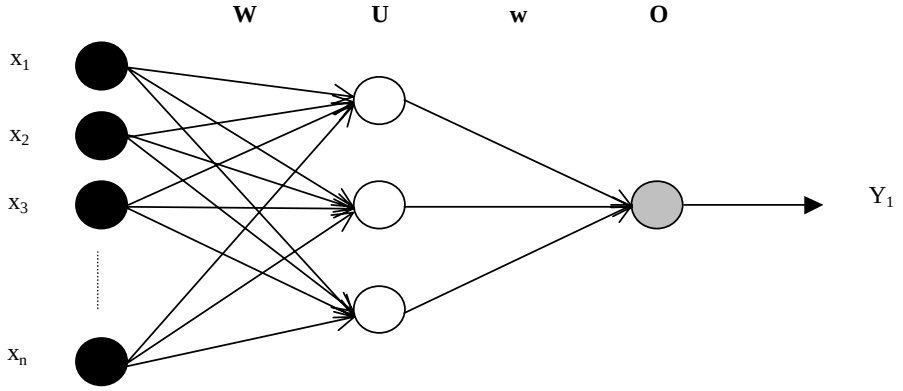
Każdą z tych funkcji można minimalizować, ze względu na parametry adaptacyjne sieci **w**, dowolną metodą znajdującą minimum lokalne lub też globalne. W większości realnych przypadków liczba węzłów, a co za tym idzie liczba parametrów adaptacyjnych, jest dość duża, stosuje się więc efektywne obliczeniowo metody znajdujące minima lokalne. Są to najczęściej różne wariacje metod gradientowych. Stąd występują różne rodzaje algorytmu propagacji wstecznej np. Rprop, Quickprop.

Zastosowanie metod gradientowy prowadzi do tego, że funkcja kosztu musi być różniczkowalna, a co za tym idzie funkcje realizowane przez poszczególne neurony również muszą być różniczkowalne. Dlatego też w wielowarstwowych sieciach nie można używać funkcji skokowych, a korzystanie z ostrych funkcji sigmoidalnych może prowadzić do dużych problemów podczas uczenia.

Poniżej przedstawiona zostanie podstawowa wersja algorytmu propagacji wstecznej błędu, w której minimalizowana jest funkcja średniokwadratowa.

5.4. Algorytm uczenia

Rozpatrzymy sieć jednokierunkową, czyli taką, w której przepływ sygnałów występuje w jednym kierunku, od wejścia do wyjścia, posiadającą dwie warstwy połączone tak jak jest to przedstawione na poniższym rysunku.



Rys 5:6 Sieć MLP

Na wejście sieci wprowadzane są *n*-wymiarowe wektory **x** losowo wybrane z ciągu treningowego. Wybór dokonywany jest w taki sposób, aby każdy z wektorów w danej serii pojawił się tylko raz. Gdy cały ciąg treningowy zostanie już pokazany, procedura zostaje powtórzona. Każda taka seria nazywana jest epoką.

Na wyjściu pojawia się żądana wartość *d*. Dla prostoty przyjmujemy tylko jedno wyjście, jednak w ogólnym przypadku na wyjściu może być dowolny wektor *k*-wymiarowy.

Sygnał wpływający do jednostek U znajdujących się w warstwie ukrytej ma postać

$$s_j = \sum_k W_{jk} x_k \tag{4.18}$$

a zatem sygnał wyjściowy jednostki U_j

$$U_j = f(s_j) = f(\sum_k W_{jk} x_k) \tag{4.19}$$

Sygnał wyjściowy jednostki w warstwie U jest sygnałem wejściowym dla warstwy O. Dlatego całkowitą funkcję, którą realizuje sieć, można napisać w postaci

$$Y = f(\sum_j w_j f(\sum_i W_{ji} x_i)) \tag{4.20}$$

Stąd średniokwadratowa funkcja kosztu ma postać

$$E = \frac{1}{2}\left(f(\sum_j w_j f(\sum_i W_{ji} x_i)) - d \right)^2 \tag{4.21}$$

Natomiast pochodna po wagach w wyraża się następującym wzorem

$$\frac{\partial E}{\partial w_j} = \left(f(\sum_j w_j U_j) - d \right) f'(\sum_j w_j U_j) U_j \tag{4.22}$$

Dla wag W

$$\frac{\partial E}{\partial W_{ji}} = \sum_i \frac{\partial E}{\partial U_j} \frac{\partial U_j}{\partial W_{ji}} = \left(f(\sum_j w_j U_j) - d \right) f'(\sum_j w_j U_j) w_j U_j' x_j \tag{4.23}$$

Przyjmując oznaczenia

$$\delta^1 = \left(f(\sum_j w_j U_j) - d \right) f'(\sum_j w_j U_j) \tag{4.24}$$

$$\delta_j^2 = \delta^1 w_j U_j' \tag{4.25}$$

Równanie to umożliwia wyznaczenie błędu δ dla danej jednostki ukrytej U_j jako funkcji błędu jednostki wyjściowej, która jest przez nią pobudzona. Współczynnikami są wagi w_j, któ-

re normalnie przekazują sygnał do przodu. W tym przypadku przenoszą one błędy δ wstecz, stąd nazwa algorytmu - propagacja wsteczna błędu.

Podstawiając powyższe wzory otrzymujemy

$$\frac{\partial E}{\partial w_j} = \delta^1 U_j \text{ oraz } \frac{\partial E}{\partial W_{ji}} = \delta^2 x_j \tag{4.26}$$

czyli wzory mają tą samą postać ale przy innej interpretacji δ, drugi parametr jest w zasadzie ten sam, bo oznacza wektor wejściowy dla danej warstwy.

W klasycznym algorytmie propagacji wstecznej zmiana wag odbywa się zgodnie ze wzorem

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \nabla E(\mathbf{w}) \tag{4.27}$$

gdzie η jest parametrem uczenia, którego wartości są z przedziału [0,1], *t* oznacza aktualny krok uczenia.

Powyższe rozumowanie można uogólnić na dowolną liczbę warstw. Modyfikacja warstw metodą wstecznej propagacji ma zawsze postać:

$$\Delta W_{ij} = \eta \delta^{wyjscie} U_{wejscie} \tag{4.28}$$

znaczenie $\delta^{wyjście}$ zależy od rozpatrywanej warstwy. Dla ostatniej warstwy połączeń określone jest wzorem δ^1, a dla wszystkich pozostałych warstw określone jest przez równanie typu δ^2.

Próg występujący w funkcji transferu może być traktowany jako waga przy wejściu *x*=1, stąd zmiana progu jest opisana tymi samymi wzorami, co zmiana wag.

W ogólności nie da się pokazać, że algorytm propagacji wstecznej jest zbieżny. Dlatego też zostały zdefiniowane różne kryteria pozwalające zakończyć uczenie. Do najbardziej znanych należą:

- uczenie przerywane jest, gdy norma euklidesowa wektora gradientu osiągnęła wartość mniejszą od zadanej wartości progowej
- uczenie przerywane jest, gdy zmiana funkcji kosztu $|\Delta E| \leq \varsigma$, gdzie ς jest z góry zadanym progiem

Zupełnie odmiennym kryterium stopu jest kryterium oparte na analizie generalizacji. Po każdej epoce (lub po określonej ich liczbie) sprawdzana jest generalizacja na zbiorze walidacyjnym. Uczenie jest przerywane, gdy błąd na zbiorze walidacyjnym zaczyna rosnąć, pomimo wciąż malejącego błędu na zbiorze treningowym.

5.5. Inicjalizacja

Uczenie sieci neuronowych jest na ogół procesem trudnym i z reguły prowadzi do utykania w minimum lokalnym. Istnieją metody pozwalające zmniejszyć prawdopodobieństwo utknięcia w minimum lokalnym np. zastosowanie algorytmu znajdującego minimum globalne (np. symulowane wyżarzanie). Są to jednak metody bardzo kosztowne i wymagają dużego doświadczenia.

Innym rozwiązaniem, pozwalającym znaleźć optymalne rozwiązanie metodami gradientowymi, jest odpowiedni dobór wstępnych wartości wag. Umożliwia to uniknięcie większości

niepożądanych minimów i jednocześnie znacznie przyspiesza proces uczenia. Niestety w ogólnym przypadku nie istnieje metoda doboru wag zapewniająca właściwy punkt startowy dla dowolnego problemu. Błędna inicjalizacja może spowodować powstawanie węzłów, których wartość aktywacji dla dowolnych wektorów wejściowych wynosi 1 lub 0. W takim przypadku pochodna funkcji sigmoidalnej jest zero, gdyż

$$f'(\mathbf{x}) = f(\mathbf{x})(1 - f(\mathbf{x})) \tag{4.29}$$

Zaproponowanych zostało wiele różnych metod inicjalizacji wag. Najprostszą spośród nich jest inicjalizacja losowa, która wybierana jest najczęściej z powodu jej prostoty oraz możliwości tworzenia różnych punktów startowych. Startując z różnych wag sieć na ogół utykać będzie w różnych minimach lokalnych, co może prowadzić do powstawanie równoważnych (w sensie minimalizowanej funkcji) rozwiązań. Występuje wiele metod losowej inicjalizacji. Różnice pomiędzy poszczególnymi metodami polegają na wyznaczaniu wag w oparciu o próbkowanie różnych rozkładów np. normalnego oraz na różnym doborze przedziału wartości, z którego wagi są losowane. Celem inicjalizacji losowej jest taki dobór wag, aby dla większości wektorów ze zbioru treningowego wzbudzenia poszczególnych węzłów były z obszaru liniowej zmienności funkcji transferu. W przypadku funkcji sigmoidalnej jest to przedział [-2,2] (przy jednostkowym skosie). Porównanie i opis różnych metod inicjalizacji losowej można znaleźć w pracy [61].

W innych metodach stosuje się np. algorytmy grupowania, czy też różne sposoby analizy statystycznej np. analizę dyskryminacyjną, które niekiedy pozwalają wystartować z punktu będącego blisko głębokiego minimum lokalnego. Taka inicjalizacja może znacznie przyspieszyć proces uczenia.

5.6. Podstawowe odmiany algorytmu propagacji wstecznej

Standardowy algorytm propagacji wstecznej ma kilka wad:

- utyka w płytkich minimach lokalnych
- wymaga ciągłych zmian parametru uczenia η
- jest wolno zbieżny.

Problemy te zostały częściowo rozwiązane poprzez zastosowanie innych odmian algorytmów gradientowych lub połączenie metody gradientowej z wiedzą heurystyczną. Poniżej przedstawione zostaną trzy podstawowe odmiany propagacji wstecznej.

5.6.1. Propagacja wsteczna z momentem

W standardowym algorytmie propagacji wstecznej błędu zmniejszenie błędu może być powolne wtedy, gdy jednocześnie współczynnik uczenia η i gradient są małe. Powolne uczenie może być również spowodowane wystąpieniem oscylacji wokół lokalnego minimum, co jest skutkiem zbyt dużego parametru uczenia. Jedną z metod pozwalających przyspieszyć zbieżność algorytmu jest metoda momentu mająca postać:

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \nabla E(\mathbf{w}) + \alpha \Delta \mathbf{w}(t) \tag{4.30}$$

parametr α jest stałą o wartość z przedziału (0,1).

Trzeci składnik powyższego równania nazywany jest momentem. Czynnik ten nadaje punktowi reprezentującemu wagi pewien moment bezwładności. Gdy kolejne gradienty wskazują ten sam kierunek, wówczas ich działanie kumuluje się i punkt przesuwa się w jednym kierunku, a jego przyrosty stają się coraz większe. Pojawienie się gradientu wymuszającego ruch w przeciwnym kierunku nie od razu spowoduje zawrócenie punktu. W sytuacji odwrotnej, gdy kolejne gradienty są przeciwnie skierowane, ich oddziaływanie częściowo się znosi i ruch punktu jest hamowany.

5.6.2. QuickProp

Ideą algorytmu quickprop [23] jest aproksymowanie lokalne funkcji błędu za pomocą funkcji kwadratowej. Na podstawie wartości wag i gradientów w dwóch punktach wyznaczana jest parabola. Kolejna zmiana wag dokonywana jest w taki sposób, aby znaleźć się w minimum tej paraboli. To prowadzi do wyrażenia na zmianę wag w postaci:

$$\Delta w_i(t+1) = \begin{cases} \dfrac{\nabla^t_{w_i} E(\mathbf{w})}{\nabla^{t-1}_{w_i} E(\mathbf{w}) - \nabla^t_{w_i} E(\mathbf{w})} \Delta w_i(t) & gdy \quad \Delta w_i(t) \neq 0 \\ \eta \nabla_{w_i} E(\mathbf{w}) & w \quad przeciwnym \quad wypadku \end{cases} \tag{4.31}$$

Jak wykazują testy w porównaniu ze zwykłym algorytmem propagacji wstecznej przyspieszenie procesu uczenia jest kilkusetkrotne.

5.6.3. Rprop

Jest to prosty algorytm heurystyczny Riedmillara i Brauna [53]. W algorytmie tym przy zmianie wag uwzględnia się jedynie znak składowej gradientu, pomijając jej wartość

$$\Delta w_i(t+1) = -\eta_i(t) \operatorname{sgn}\left(\frac{\partial E(\mathbf{w}(t))}{\partial w_i} \right) \tag{4.32}$$

Współczynnik uczenia jest dobierany w każdym cyklu dla każdej wagi indywidualnie na podstawie zmian wartości gradientu

$$\eta_i(t) = \begin{cases} \min(a\eta_i(t-1), \eta_{\max}) & wtedy \quad gdy \quad \nabla^t_{w_i} E(\mathbf{w}) \nabla^{t-1}_{w_i} E(\mathbf{w}) > 0 \\ \max(b\eta_i(t-1), \eta_{\min}) & wtedy \quad gdy \quad \nabla^t_{w_i} E(\mathbf{w}) \nabla^{t-1}_{w_i} E(\mathbf{w}) < 0 \\ \eta_i(t-1) & w \quad pozostaych \quad przypadkach \end{cases} \tag{4.33}$$

gdzie a,b są stałymi a=1.2, b=0.5, $\eta_{\min}$, $\eta_{\max}$ oznaczają odpowiednio minimalną i maksymalną wartość współczynnika uczenia, równą $\eta_{\min} = 10^{-6}$, $\eta_{\max} = 50$.

Algorytm Rprop umożliwia znaczne przyspieszenie procesu uczenia w tych obszarach, gdzie nachylenie funkcji celu jest niewielkie. Strategia zmiany wag zakłada ciągły wzrost współczynnika uczenia, jeśli w dwóch kolejnych krokach znak gradientu jest taki sam, natomiast jego redukcję, gdy znak ten jest różny.

Sieci neuronowe, w których korzysta się z minimalizacji metodą gradientową, zawsze używają ciągłej funkcji kosztu. Mimo to można stosować sieci neuronowe ze średniokwadratową funkcją błędu do problemów klasyfikacyjnych, w których funkcja błędu jest nieciągła. Jak wykazał White [50] możliwa jest aproksymacja prawdopodobieństw klas a posteriori za pomocą MLP podczas minimalizowania błędu średniokwadratowego. Pod warunkiem, że zbiór treningowy jest wystarczająco duży i w trakcie uczenia sieć nie utknie w jakimś lokalnym minimum.

Błąd średniokwadratowy po odrzuceniu członu związanego z szumem danych (1.13) dla problemu L klas można przedstawić w następujący sposób

$$\sum_{i=1}^{L}(y_i(\mathbf{x})-<d_i\mid\mathbf{x}>)^2 \tag{4.34}$$

ale

$$<d_i\mid\mathbf{x}>=E(d_i\mid\mathbf{x})=\sum_{j=1}^{L}d_i p(C_j\mid\mathbf{x})=p(C_i\mid\mathbf{x}) \tag{4.35}$$

Powyższe równania pokazują, że gdy funkcja średniokwadratowa jest minimalizowana to wyjście sieci estymuje prawdziwe prawdopodobieństwa Bayesowskie. W przypadku wieloklasowym istnieje wiele możliwości analizowania wyjść sieci np. uznaje się, że dany wektor jest klasyfikowany do danej klasy tylko wtedy, gdy odpowiedni węzeł wyjściowy ma wzbudzenie powyżej 0.5 (przy zastosowaniu funkcji sigmoidalnej). Jednak, ponieważ węzły wyjściowe estymują prawdopodobieństwa a posteriori, to z Bayesowskiego punktu widzenia klasyfikator optymalny wybiera klasę C_k wtedy, gdy:

$$p(C_k\mid\mathbf{x})>p(C_i\mid\mathbf{x}),\ i=1,...,L;i\neq k \tag{4.36}$$

Niestety tak jest tylko przy założeniu, że zbiór treningowy jest nieskończenie duży. W praktyce natomiast zbiór jest skończony, dlatego też zamiast minimalizowania przestawionej funkcji błędu minimalizowana jest funkcja tej postaci

$$\sum_{j=1}^{N}\left(\sum_{i=1}^{L}(y_i(x_j)-d_i^j)^2\right) \tag{4.37}$$

Minimalizowanie powyższej funkcji prowadzi do tego, że dokładność estymacji Bayesowskich prawdopodobieństw zależy od liczby danych jak również od tego czy jest równomierny wkład wektorów z poszczególnych klas do zbioru treningowego. Poza tym istnieje również założenie, że uczona sieć jest wystarczająco złożona, aby dokładnie estymować Bayesowskie prawdopodobieństwa.. W przypadku, gdy tak nie jest dokładność estymacji zmniejsza się.

Sieć najlepiej estymuje Bayesowskie prawdopodobieństwa w obszarach, w których występuje duże prawdopodobieństwo co najmniej jednej klasy, a dużo gorzej w miejscach, w których prawdopodobieństwo wystąpienia którejkolwiek z klas jest małe. Jest to właśnie konsekwencja użycia funkcji średniokwadratowej, gdyż dużo punktów występuje w obszarach o wysokim prawdopodobieństwie, a mało w obszarach małego prawdopodobieństwa. Mała liczba punktów z danego obszaru powoduje mały wpływ na całkowitą funkcję błędu, przez co obszar taki jest źle odwzorowywany. W wyniku uczenia sieci można więc otrzymać bardzo słabą estymację pomimo dobrej minimalizacji funkcji średniokwadratowej. Jest to powód, dla którego w problemach klasyfikacyjnych sieci neuronowe nie zawsze sprawują się dobrze. Symulacje pokazujące zależność estymacji od złożoności sieci oraz od liczby danych można znaleźć w pracy [50].

5.8. Obcinanie wag

W koncepcji tej sieć w trakcie uczenia sama usuwa połączenia, które nie są wzmacniane. Najprostsza metoda polega na zastosowaniu w funkcji kosztu dodatkowego członu wymuszającego małe wartości wag.

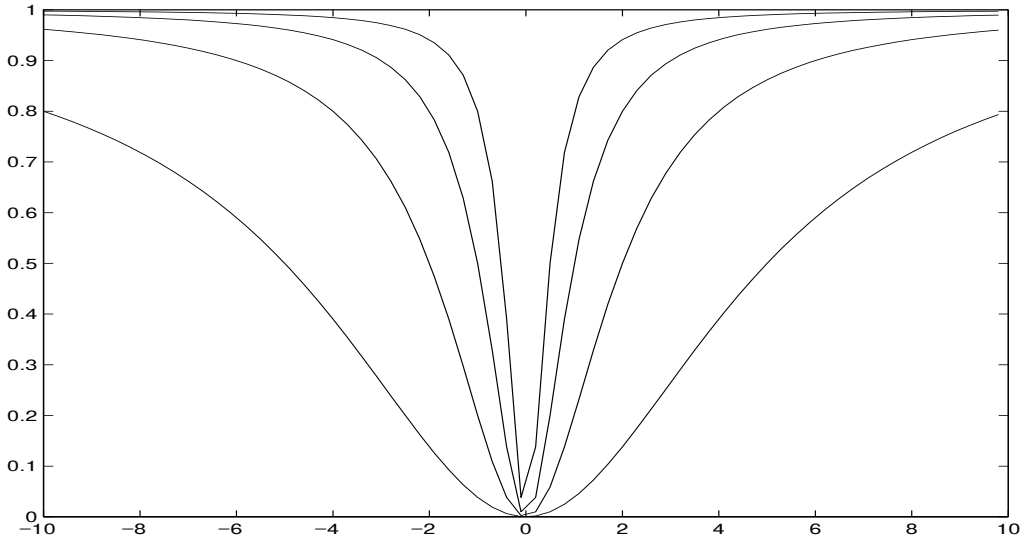
$$E = \sum_k (y(\mathbf{X}_k; \mathbf{w}) - d)^2 + \gamma \sum_i \sum_{j>i} w_{ij}^2 \tag{4.38}$$

Pierwszy człon ma standardową postać dla algorytmu propagacji wstecznej błędu natomiast drugi człon jest członem kary (członem regularyzacyjnym) wymuszającym wagi o małych wartościach. Oczywiście za pomocą parametru γ można zmieniać wpływ członu regularyzacyjnego na funkcję kosztu.

Taka postać członu kary prowadzi do obniżania wartości wszystkich wag. Jest to efekt nie zawsze pożądany. Sytuację tą można zmienić wprowadzając człon zdefiniowany przez Weigenda [28] takiej postaci

$$\sum_i \frac{(w_i / w_0)^2}{1 + (w_i / w_0)^2} \tag{4.39}$$

Sumowanie przebiega po wszystkich wagach w_i sieci. Parametr w_0 jest parametrem swobodnym, którego wielkość decyduje, w zależności od wartości wag w_i, o stopniu kary np. gdy $|w_i| << w_0$ wówczas człon regularyzacyjny jest bardzo mały, a gdy $|w_i| >> w_0$ człon regularyzacyjny osiąga swoje maksimum, co oznacza, że waga jest potrzebna.



Rys 5:7 Wykres członu regularyzacyjnego Weigenda dla wartości w_0: 0.5,1,2.5

Człon regularyzacyjny powoduje usunięcie tych wag w sieci, które wydają się zbędne tzn. których wartość jest bardzo mała.

5.9. OBD (Optimal Brain Damage)

OBD [12] pozwalająca zmniejszyć złożoność sieci neuronowej (liczbę parametrów adaptacyjnych), która została już nauczona. Metoda ta oparta jest na konstrukcji lokalnego modelu funkcji błędu i w oparciu o ten model dokonuje analitycznej predykcji efektów związanych z perturbacją wektora parametrów.

Aby zbadać wpływ perturbacji $\delta\mathbf{w}$ wektora parametrów $\mathbf{w}$ na zmianę funkcji błędu rozwija się funkcję błędu w szereg Taylora. Stąd

$$\delta E = \sum_i \frac{\partial E}{\partial w_i}\delta w_i + \frac{1}{2}\sum_i \frac{\partial^2 E}{\partial w_i \partial w_i}\delta w_i^2 + \frac{1}{2}\sum_{i\neq j} \frac{\partial^2 E}{\partial w_i \partial w_j}\delta w_i \delta w_j + O(\| \delta w \|^3) \qquad (4.40)$$

Celem jest znalezienie takiego zbioru parametrów, których usunięcie z sieci spowoduje najmniejszy wzrost funkcji błędu E. Niestety w tej postaci problem jest praktycznie nie do rozwiązania, ponieważ przy dużej ilości parametrów adaptacyjnych sieci liczba elementów hesjanu jest ogromna, dlatego też wprowadza się pewne uproszczenie. Dokonuje się przybliżenia „diagonalnego", które zakłada, że perturbacja funkcji błędu spowodowana usunięciem niektórych wag, jest sumą perturbacji związanych z usunięciem każdego parametru z osobna. Dzięki temu trzeci człon w równaniu może zostać pominięty. Ponieważ usuwanie parametrów następuje po nauczeniu sieci, czyli w jakimś minimum lokalnym, to można również pominąć pierwszy człon równania. Czwarty człon równania może zostać pominięty, gdyż zakładamy nieduże perturbacje wag. Stąd zostaje

$$\delta E = \frac{1}{2}\sum_i \frac{\partial^2 E}{\partial w_i \partial w_i}\delta w_i^2 = \sum_i h_{ii}\delta w_i^2 \qquad (4.41)$$

Cała procedura OBD przeprowadzana jest w następujący sposób:

1. wybierz odpowiednią architekturę sieci
2. ucz sieć tak długo, aż uzyskane zostanie rozsądne rozwiązanie
3. wylicz drugą pochodną dla każdej z wag
4. dla każdej wagi wylicz współczynnik h_{kk}
5. usuń wagi, których współczynniki h_{kk} maja wartości poniżej zadanego progu, jeżeli takich nie ma to kończ
6. powrót do punktu 2.

Po każdym usunięciu wag wymagane jest douczanie. Istnieje inny algorytm OBS (Optimal Brain Surgeon) oparty na podobny założeniach, co OBD, w którym douczanie nie jest konieczne, gdyż wraz z każdym usunięciem jakieś wagi pozostałe wagi są modyfikowane w taki sposób, aby zminimalizować wzrost funkcji błędu. Szczegółowy opis tego algorytmu można znaleźć w pracach [25],[27].

5.10. Sieci konstruktywistyczne

Drugą metodą automatycznego określania architektury jest tworzenie sieci od zera. W trakcie uczenia dostawiane są w warstwie ukrytej kolejne węzły, lub też dostawiane są kolejne warstwy ukryte. Ten typ algorytmów określany jest mianem algorytmów konstruktywistycznych. Do najbardziej znanych algorytmów konstruktywistycznych dla sieci typu propagacji wstecznej należy korelacja kaskadowa Fahlmana [22].

Na początku nie istnieje żaden węzeł ukryty, każdy węzeł wyjściowy połączony jest ze wszystkimi węzłami wejściowymi. Istnieje więc tylko jedna warstwa neuronów, która może być uczona dowolnym algorytmem. Fahlman w swej pracy użył algorytmu quickprop. Sieć uczona jest tak długo, aż dalsze uczenie nie powoduje malenia błędu. Gdy po pewnej liczbie epok (kontrolowanej przez tzw. parametr cierpliwości), nie nastąpiła znacząca redukcja błędu (również zdefiniowana przez użytkownika), uczenie jest przerywane. Jeżeli błąd na całym zbiorze treningowym jest mniejszy niż z góry założona tolerancja, proces uczenia jest przerywany, w przeciwnym wypadku dostawiany jest nowy węzeł. Po dostawieniu nowego węzła wszystkie wagi dochodzące do jednostek wyjściowych są uczone, natomiast wszystkie pozostałe zostają zamrożone. Ponieważ w wyniku zamrożenia sygnał wejściowy jednostek ukrytych dla poszczególnych wektorów nie ulega zmianie, to aktywacja tych jednostek może być przeliczona tylko raz dla całego zbioru treningowego i zapamiętana, a następnie wykorzystana podczas trenowanie sieci, co pozwala znacznie przyspieszyć proces uczenia.

Nowy węzeł łączony jest ze wszystkimi węzłami wejściowymi, wszystkimi istniejącymi węzłami ukrytymi oraz wyjściowymi, przez co powstaje struktura kaskadowa. Dla każdego nowego węzła wyznaczane są wagi połączeń pomiędzy wszystkimi istniejącymi neuronami a nowym węzłem oprócz węzłów wyjściowych. Wagi te wyznaczane są przez maksymalizację korelacji pomiędzy wyjściem neuronu zwanego kandydatem, a błędem istniejącej sieci neuronowej. Korelacja ta zdefiniowana jest w następujący sposób

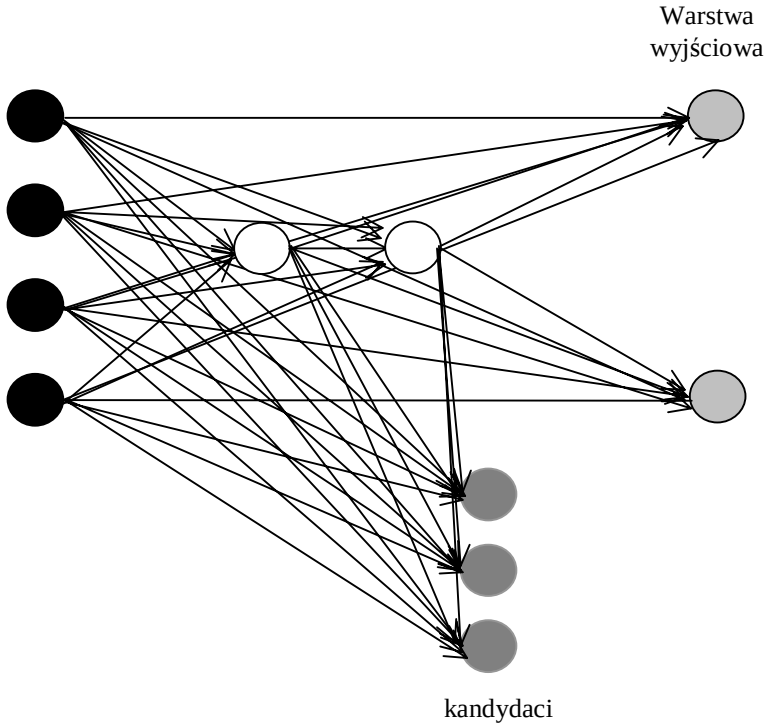
$$S = \sum_o |\sum_i^N (h^i - \bar{h})(E_o^i - \overline{E_o})| \tag{4.42}$$

gdzie o jest indeksem wyjść sieci, i to wskaźnik trenowanego wzorca, h jest wyjściem kandydata, $\bar{h}$ jest średnią obliczoną ze wszystkich wartości pojawiających się na wyjściu kan-

dydata, E_o^i błąd sieci dla *i*-tego wzorca i *o*-tego wyjścia sieci, $\overline{E_o}$ średni błąd pojawiający się na wyjściu *o*.

Zamiast pojedynczego kandydata w celu przyspieszenia procesu uczenia używa się większej liczby kandydatów. Każdy kandydat połączony jest zawsze ze wszystkimi węzłami ukrytymi i wejściowymi. Na początku wagi wybierane są losowo. Do wszystkich kandydatów podawany jest ten sam błąd E_o^i. Po wytrenowaniu kandydatów wybierany jest ten, który ma największą korelację *S*. Dzięki temu, że używa się wielu kandydatów można zredukować możliwość tworzenia złej jednostki ukrytej. W przypadku obliczeń za pomocą komputerów równoległych można znacznie przyspieszyć proces treningowy.

Przykład sieci korelacji kaskadowej z dwoma neuronami ukrytymi i trzema kandydatami przedstawiony jest na poniższym rysunku.



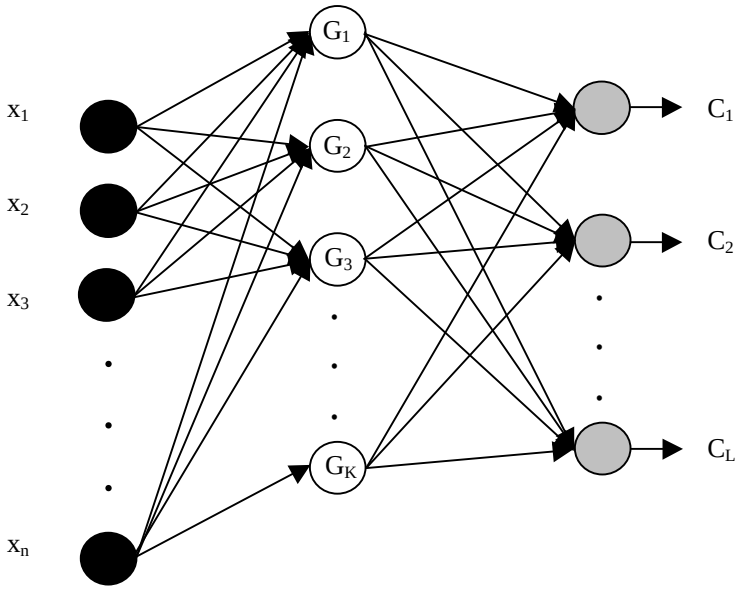
Rys 5:8 Sieć korelacji kaskadowej

6. Sieci RBF

Sieci MLP bazują na neuronach, które realizują nieliniową funkcję sygnału wejściowego. Sieci RBF (Radial Basis Functions) reprezentują drugą klasę sieci, w których funkcja transferu dla węzłów ukrytych jest funkcją odległości pomiędzy wektorem wejściowym a wektorem prototypowym.

6.1. Architektura sieci RBF

W przeciwieństwie do sieci MLP architektura sieci RBF jest znacznie prostsza i zawsze składa się tylko z dwóch warstw neuronów: ukrytej i wyjściowej.



Rys 6:1 Architektura sieci RBF

Neurony w warstwie ukrytej połączone są w pełni z neuronami wejściowymi. Liczba węzłów wejściowych równa jest wymiarowi wektora wejściowego, natomiast liczba węzłów w warstwie ukrytej jest mniejsza lub równa liczbie wektorów w zbiorze treningowym. Neurony warstwy wyjściowej, których liczba równa jest liczbie klas w zbiorze treningowym, realizują ważoną kombinację wyjść neuronów warstwy ukrytej. Funkcję realizowaną przez sieć RBF z jednym węzłem wyjściowym można przedstawić w następujący sposób

$$y(\mathbf{x}) = \sum_{i}^{K} w_i G(\mathbf{x}; \mathbf{D}_i) \tag{5.1}$$

gdzie $G(\mathbf{x}; \mathbf{D}_i) = G(\| \mathbf{x} - \mathbf{D}_i \|)$ jest *i*-tą radialną funkcją bazową, czyli funkcją zmieniającą się radialnie wokół wybranego centrum $\mathbf{D}_i$, natomiast $\| \cdot \|$ jest zazwyczaj normą euklidesową. Stosowanie funkcji radialnych jako funkcji bazowych ma swoje uzasadnienie zarówno z punktu widzenia teorii interpolacji, jak również teorii regularyzacji. Zagadnienie to zostało szczegółowo opisane w książkach [5],[28],[45], dlatego tu przedstawione zostaną tylko te elementy, które istotne są z punktu widzenia rozwijanego w dalszej części pracy modelu FSM.

W teorii regularyzacji do standardowej funkcji kosztu (np. takiej, jak w równaniu (1.7)) dodaje się człon stabilizujący, pozwalający na uwzględnienie dodatkowych warunków, które spełniać powinna funkcja aproksymująca. Jeśli te warunki są niezmiennicze względem translacji i rotacji w przestrzeni cech, funkcje aproksymujące powinny mieć postać radialną.

Znanych jest wiele funkcji radialnych, np. funkcja gaussowska (5.2), odwrotna f. potęgowa (5.3), potęgowa (5.4), funkcja „cienkiej płytki" (5.5) i funkcja stożkowa (5.6)

$$G(\mathbf{x};\mathbf{D},\sigma) = e^{-\left(\frac{\|\mathbf{x}-\mathbf{D}\|}{\sigma}\right)^2} \tag{5.2}$$

$$G(\mathbf{x};\mathbf{D},c,\alpha) = \frac{1}{(c^2+\|\mathbf{x}-\mathbf{D}\|^2)^{\alpha}} \quad \alpha > 0 \tag{5.3}$$

$$G(\mathbf{x};\mathbf{D},c,\beta) = \left(c^2+\|\mathbf{x}-\mathbf{D}\|\right)^{\beta} \quad 0 < \beta < 1 \tag{5.4}$$

$$G(\mathbf{x};\mathbf{D},c) = \left(c\,\|\mathbf{x}-\mathbf{D}\|\right)^2 \ln\left(c\,\|\mathbf{x}-\mathbf{D}\|\right) \tag{5.5}$$

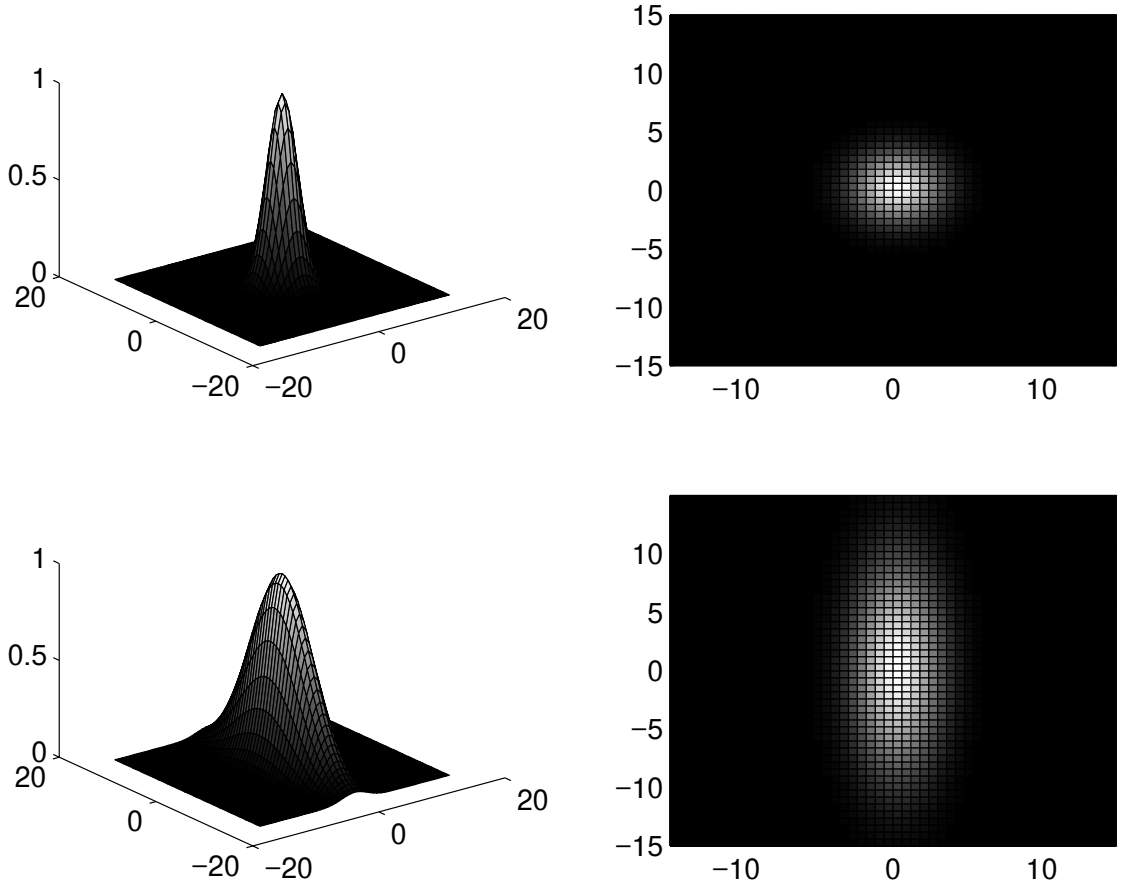
$$G(\mathbf{x};\mathbf{D}) = \|\mathbf{x}-\mathbf{D}\| \tag{5.6}$$

Najczęściej używaną funkcją jest funkcja gaussowska (5.2), która jest funkcją zlokalizowaną, czyli $G(\mathbf{x};\mathbf{D},\sigma)\to 0$ dla $|\mathbf{x}|\to\infty$, maksimum osiąga natomiast dla $\mathbf{D}=\mathbf{x}$. Oznacza to, że niezerowa wartość wzbudzenia neuronu realizującego funkcję gaussowską odbywa się tylko w pobliżu jego centrum **D**.

Funkcja gaussowska jako jedyna funkcja radialna jest faktoryzowalna, czyli wielowymiarowa radialna funkcja bazowa może zostać przedstawiona jako iloczyn funkcji jednowymiarowych

$$G(\mathbf{x};\mathbf{D},\sigma) = e^{-\left(\frac{\|\mathbf{x}-\mathbf{D}\|}{\sigma}\right)^2} = \prod_{i=1}^{n} e^{-\left(\frac{x_i-D_i}{\sigma_i}\right)^2} \tag{5.7}$$

Własność faktoryzowalności pozwala traktować każdy z wymiarów niezależnie, co umożliwia niezależną zmianę parametrów adaptacyjnych dla każdego z wymiarów i jednocześnie pozwala tylko raz wyliczyć funkcję eksponencjalną.



Rys 6:2 Wykres dwuwymiarowej funkcji gaussowskiej z jednakowym rozmyciem dla obu wymiarów oraz z różnymi rozmyciami

Zastosowanie funkcji gaussowskiej (5.7) umożliwia tworzenie hipereliptycznych obszarów decyzyjnych za pomocą jednego węzła, przy czym nie ma możliwości obrotu tych obszarów, czyli główne osie hiperelipsy są zgodne z osiami układu współrzędnych.

Oprócz funkcji gaussowskiej przedstawionej we wzorze (5.2) w sieciach RBF stosowana jest również jej uogólniona postać

$$G(\mathbf{x};\mathbf{D},\sigma) = e^{-\left(\frac{1}{2}(\mathbf{x}-\mathbf{D})^T \Sigma (\mathbf{x}-\mathbf{D})\right)} \tag{5.8}$$

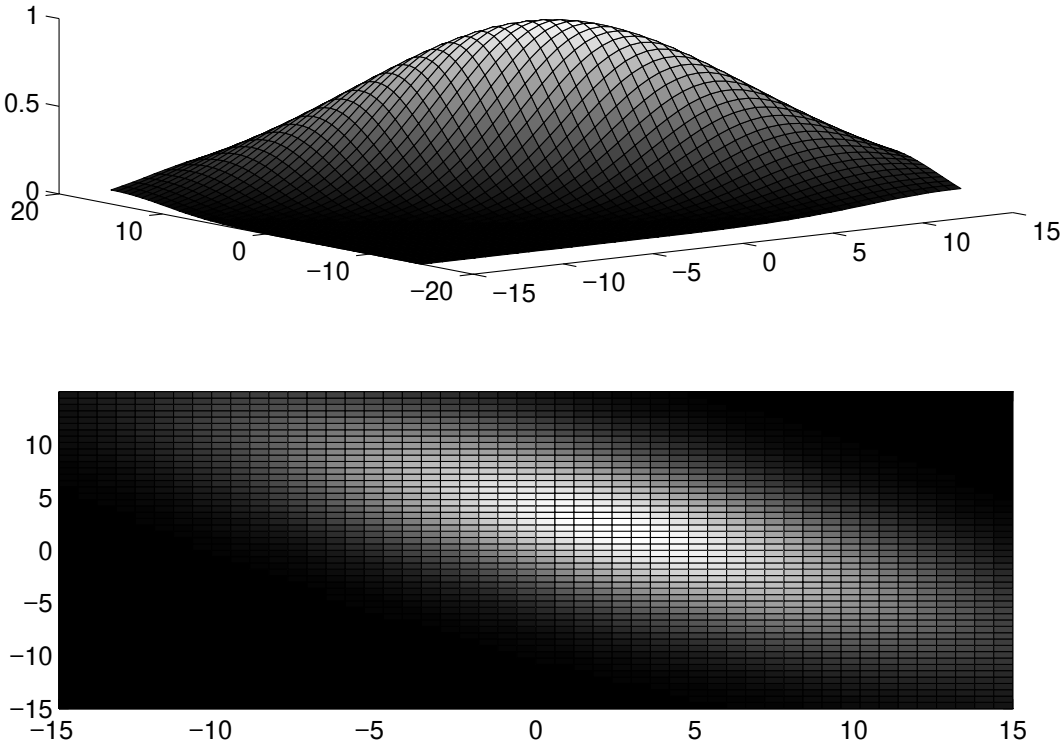
gdzie Σ to macierz kowariancji postaci

$$\Sigma = E\left[(\mathbf{x}-\mathbf{D})(\mathbf{x}-\mathbf{D})^T\right] \tag{5.9}$$

natomiast $E[\ ]$ jest operatorem wartości oczekiwanej.

W wyniku zastosowania funkcji (5.8) powstają hipereliptyczne obszary decyzyjne, które dzięki występowaniu pozadiagonalnych elementów w macierzy Σ mogą być obrócone w dowolny sposób. Niestety poważną wadą tej funkcji jest występowanie bardzo wielu parametrów związanych z macierzą kowariancji, których liczba rośnie kwadratowo ze wzrostem liczby wymiarów. Przy problemach z dużą liczbą wymiarów n powstaje macierz n^2, którą trzeba jeszcze odwrócić, co wprowadza dodatkowe utrudnienia, gdyż macierz ta może być osobliwa.

Stosowanie tej funkcji jest więc bardzo kosztowne. Dlatego w realnych zastosowaniach dużo częściej stosuje się funkcję (5.7) niż (5.8).



Rys 6:3 Obrócona funkcja gaussowska

6.2. Algorytm uczenia

Celem uczenia sieci RBF jest znalezienie parametrów adaptacyjnych tj. wag **w**, parametrów funkcji bazowych takich jak: położenie **D** oraz w przypadku zastosowania funkcji gaussowskiej rozmycia σ, które minimalizują funkcję kosztu. Podobnie jak w przypadku sieci MLP najczęściej stosuje się tu funkcję średniokwadratową

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{l=1}^{L}\left(\sum_{j=1}^{K} w_{lj} G_j^i - d_l^i\right)^2 \tag{5.10}$$

6.2.1. Wyznaczanie wag

Załóżmy, że parametry funkcji bazowej zostały już znalezione, pozostaje więc wyznaczyć wagi. Różniczkując funkcję błędu ze względu na wagi otrzymuje się równanie, które w reprezentacji macierzowej ma postać

$$\left(\mathbf{G}^T\mathbf{G}\right)\mathbf{w}^T = \mathbf{G}^T\mathbf{d} \tag{5.11}$$

gdzie macierz **G** zawiera elementy G_j^i i ma wymiar $n \times L$,

$$\mathbf{G} = \begin{bmatrix} G(\mathbf{x}_1;\mathbf{D}_1) & G(\mathbf{x}_1;\mathbf{D}_2) & \cdots & G(\mathbf{x}_1;\mathbf{D}_K) \\ G(\mathbf{x}_2;\mathbf{D}_1) & G(\mathbf{x}_2;\mathbf{D}_2) & \cdots & G(\mathbf{x}_2;\mathbf{D}_K) \\ \vdots & \vdots & & \vdots \\ G(\mathbf{x}_N;\mathbf{D}_1) & G(\mathbf{x}_N;\mathbf{D}_2) & \cdots & G(\mathbf{x}_N;\mathbf{D}_K) \end{bmatrix} \tag{5.12}$$

w macierz wag o wymiarze $L \times K$, **d** jest macierzą zawierającą elementy d_l^i o wymiarze $n \times L$. Stąd iloczyn macierzy $\mathbf{G}^T\mathbf{G}$ jest macierzą $K \times K$ i w przypadku, gdy jest to macierz nieosobliwa rozwiązanie jest następujące

$$\mathbf{w}^T = \mathbf{G}^{\dagger}\mathbf{d} \tag{5.13}$$

gdzie $\mathbf{G}^{\dagger}$ jest tzw. macierzą pseudoodwrotną postaci

$$\mathbf{G}^{\dagger} = (\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T \tag{5.14}$$

W przypadku, gdy liczba neuronów ukrytych jest duża mogą występować problemy z odwracaniem macierzy, poza tym z powodu zbyt dużej liczby parametrów adaptacyjnych sieć dokonuje zbyt szczegółowo odwzorowania, co ma znaczenie przy końcowej generalizacji. W celu kontrolowania gładkości odwzorowania realizowanego przez sieć, a co za tym idzie liczby parametrów adaptacyjnych, podobnie jak w sieciach MLP stosuje się regularyzację. W związku z tym do funkcji błędu dodawany jest człon regularyzacyjny

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{l=1}^{L}\left(\sum_{j=1}^{K} w_{lj}G_j^i - d_l^i\right)^2 + \lambda \| \mathbf{P}y \|^2 \tag{5.15}$$

gdzie λ jest parametrem regulującym wpływ członu regularyzacyjnego na funkcję błędu, **P** operatorem różniczkowym.

W wyniku minimalizacji funkcji (5.15) ze względu na wagi otrzymuje się rozwiązanie [28] postaci

$$\left(\mathbf{G}^T\mathbf{G} + \lambda\mathbf{G}_0\right)\mathbf{w} = \mathbf{G}^T\mathbf{d} \tag{5.16}$$

gdzie

$$\mathbf{G}_0 = \begin{bmatrix} G(\mathbf{D}_1;\mathbf{D}_1) & G(\mathbf{D}_1;\mathbf{D}_2) & \cdots & G(\mathbf{D}_1;\mathbf{D}_K) \\ G(\mathbf{D}_2;\mathbf{D}_1) & G(\mathbf{D}_2;\mathbf{D}_2) & \cdots & G(\mathbf{D}_2;\mathbf{D}_K) \\ \vdots & \vdots & & \vdots \\ G(\mathbf{D}_K;\mathbf{D}_1) & G(\mathbf{D}_K;\mathbf{D}_2) & \cdots & G(\mathbf{D}_K;\mathbf{D}_K) \end{bmatrix} \tag{5.17}$$

Oczywiście w przypadku, gdy parametr regularyzacyjny λ=0 otrzymuje się rozwiązanie (5.13).

6.2.2. Wyznaczanie parametrÛw adaptacyjnych funkcji bazowych

Aby znaleźć wagi w sposób przedstawiony powyżej, trzeba najpierw znaleźć parametry adaptacyjne funkcji bazowych oraz wyznaczyć ich liczbę. Można wyróżnić dwie strategie wyznaczania położeń funkcji radialnych: bez nadzoru i pod nadzorem.

Strategie bez nadzoru polegają na tym, że przy ustalaniu położeń funkcji bazowej bierze się pod uwagę tylko informacje zawarte w wektorach wejściowych i zupełnie pomija się informację wyjściową. Najprostszą metodą wyznaczenia położeń funkcji bazowych jest metoda wyznaczania centrów w sposób losowy. Położenia wyznaczane są poprzez losowanie zadanej liczby wektorów z ciągu treningowego. W przypadku funkcji gaussowskiej (5.7) występują dodatkowe parametry, które obliczane są w następujący sposób

$$\sigma = \frac{t}{\sqrt{2M}} \tag{5.18}$$

gdzie t jest maksymalną odległością pomiędzy wybranymi centrami, natomiast M liczbą funkcji bazowych.

Taki wybór odchylenia standardowego σ pozwala uniknąć tworzenia funkcji gaussowskiej, która byłaby zbyt wąska lub zbyt szeroka. Tego typu metoda, przy próbie osiągnięcia odpowiednio wysokiej dokładności na danych treningowy, może prowadzić niestety do powstawania zbyt dużej liczby funkcji bazowych. Dlatego też stosowana jest raczej jako punkt startu dla metod iteracyjnych.

Kolejną metodą jest metoda polegająca na wykorzystaniu algorytmów klasteryzacyjnych [41]. Parametry funkcji bazowych wyznaczane są poprzez metodę samoorganizacji występującą w algorytmie k-średnich. W wyniku klasteryzacji otrzymuje się skupiska, w środku których umiejscawiane są funkcje bazowe. Rozmycia funkcji gaussowskiej mogą zostać wyliczone na podstawie kowariancji punktów należących do danego skupiska.

Strategia pod nadzorem polega na wykorzystaniu informacji wyjściowych przy ustalaniu wszystkich parametrów sieci, czyli: położeń ewentualnie rozmyć funkcji bazowych oraz wag. W tym przypadku w trakcie uczenia zmieniają się równocześnie wszystkie parametry. Zmiana parametrów odbywa się poprzez gradientową minimalizację funkcji (5.10). Stąd zmiana wag dana jest następującym wzorem

$$w_i(t+1) = w_i(t) - \eta_1 \frac{\partial E}{\partial w_i(t)} \tag{5.19}$$

zmiana położenia funkcji bazowej

$$\mathbf{D}_i(t+1) = \mathbf{D}_i(t) - \eta_2 \frac{\partial E}{\partial \mathbf{D}_i(t)} \tag{5.20}$$

i rozmycia

$$\sigma_i(t+1) = \sigma_i(t) - \eta_3 \frac{\partial E}{\partial \sigma_i(t)} \tag{5.21}$$

Zastosowanie metody optymalizacyjnej powoduje niestety znaczne spowolnienie procesu uczenia w porównaniu do metod, w których położenia węzłów są ustalone.

6.3. Probabilistyczna interpretacja węzłów ukrytych w sieci RBF

W problemach klasyfikacyjnych, z punktu widzenia teorii prawdopodobieństwa, celem jest modelowanie prawdopodobieństwa a posteriori każdej z klas występujących w zbiorze treningowym. Prawdopodobieństwa te mogą zostać wyrażone w następujący sposób

$$p(C_j \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid C_j)\, p(C_j)}{p(\mathbf{x})} \tag{5.22}$$

gdzie $p(C_j)$ jest prawdopodobieństwem a priori wystąpienia klasy C_j, natomiast $p(\mathbf{x}|C_j)$ jest prawdopodobieństwem modelowanym przez odpowiednie funkcje bazowe. W przypadku M funkcji bazowych modelujących rozkład danej klasy prawdopodobieństwo $p(\mathbf{x}|C_j)$ ma postać

$$p(\mathbf{x} \mid C_j) = \sum_{i=1}^{M} p(\mathbf{x} \mid i)\, p(i \mid C_j) \tag{5.23}$$

Stąd $p(\mathbf{x})$

$$p(\mathbf{x}) = \sum_{j} p(\mathbf{x} \mid C_j)\, p(C_j) = \sum_{i=1}^{M} p(\mathbf{x} \mid i)\, p(i) \tag{5.24}$$

gdzie $p(i)$ zdefiniowane jest jako

$$p(i) = \sum_{j} p(i \mid C_j)\, p(C_j) \tag{5.25}$$

Prawdopodobieństwo a posteriori ma więc postać

$$p(C_j \mid \mathbf{x}) = \frac{\sum_{i=1}^{M} p(\mathbf{x} \mid i)\, p(i \mid C_j)\, p(C_j)}{\sum_{s=1}^{M} p(\mathbf{x} \mid s)\, p(s)} \frac{p(i)}{p(i)} = \sum_{i=1}^{M} w_{ji} \phi_i(\mathbf{x}) \tag{5.26}$$

Równanie (5.26) przedstawia sieć z radialnymi funkcjami bazowymi, w której znormalizowane funkcje bazowe dane są następującym wzorem

$$\phi_i(\mathbf{x}) = \frac{p(\mathbf{x} \mid i)\, p(i)}{\sum_{s=1}^{M} p(\mathbf{x} \mid s)\, p(s)} = p(i \mid \mathbf{x}) \tag{5.27}$$

natomiast wagi

$$w_{ij} = \frac{p(i \mid C_j)\, p(C_j)}{p(i)} = p(C_j \mid i) \tag{5.28}$$

Jak wynika z równania (5.27) aktywacja funkcji bazowej może być interpretowana jako prawdopodobieństwo a posteriori tego, że w przestrzeni wejściowej istnieją odpowiednie cechy (charakteryzowane przez daną funkcje bazową). Wagi natomiast mogą być interpretowane jako prawdopodobieństwo a posteriori przynależności do klas przy istnieniu odpowiednich cech.

Ponieważ w sieci RBF podobnie jak w sieciach MLP minimalizowana jest funkcja średniokwadratowa, to węzły wyjściowe mają interpretacje prawdopodobieństw a posteriori odpowiadających im klas.

6.4. Sieć RAN

Osiągnięcie dobrej generalizacji w sieciach neuronowych jest możliwe przy odpowiednim doborze liczby parametrów adaptacyjnych, co w sieciach RBF wyraża się liczbą węzłów ukrytych. Zbyt mała liczba węzłów ukrytych powoduje, że sieć nie może nauczyć się ciągu treningowego, zbyt duża może prowadzić do przeuczenia a co za tym idzie słabej generalizacji. W przypadku zbyt dużej liczby parametrów jednym z rozwiązań, które pozwala uzyskać sieć o dobrej generalizacji jest zastosowanie regularyzacji. Innym rozwiązaniem doboru liczby parametrów adaptacyjnych jest konstruktywistyczny algorytm uczenia. Budowanie sieci polega na dodawaniu, lub likwidowaniu węzłów wtedy, gdy jest to konieczne (określenie warunku konieczności jest zazwyczaj kluczowym elementem takiego algorytmu), lub jeżeli liczba węzłów nie ulega zmianie następuje adaptacja istniejących parametrów sieci.

Podobnie jak w przypadku sieci MLP, gdzie przykładem takiego algorytmu był algorytm korelacji kaskadowej, w sieciach RBF również istnieją takie algorytmy a do najbardziej znany należy algorytm RAN (Resource-Allocating Network) J. Platta [47].

Architektura sieci RAN jest taka sama jak sieci RBF, czyli występuje jedna warstwa ukryta i warstwa wyjściowa. Dla prostoty przedstawiony zostanie algorytm tej sieci dla jednego węzła wyjściowego. Funkcją transferu dla węzłów ukrytych jest funkcja gaussowska (5.7). Sieć natomiast realizuje następującą funkcję

$$y(\mathbf{x}) = \sum_j w_j G_j + \theta \tag{5.29}$$

Na początku uczenia nie ma żadnego węzła ukrytego. W momencie, gdy na wejście sieci podawane są wektory treningowe sieć zapamiętuje niektóre z nich, poprzez utworzenie węzła, którego położenie równe jest położeniu zapamiętywanego wektora treningowego.

$$\mathbf{D} = \mathbf{x} \tag{5.30}$$

Występują dwa warunki określające dostawienie nowego węzła. Nowy węzeł dostawiany jest wtedy, gdy aktualny wektor wejściowy jest daleko od istniejących węzłów

$$\| \mathbf{x} - \mathbf{D}_{najbliższy} \| > \delta(t) \tag{5.31}$$

oraz, gdy różnica pomiędzy żądanym wyjściem a wyjściem otrzymywanym z sieci jest wystarczająco duża

$$|| d - y(\mathbf{x}) ||> \varepsilon \tag{5.32}$$

gdzie ε jest żądaną dokładnością dopasowania, natomiast $\delta(t)$ jest skalą rozdzielczości dopasowania sieci dla epoki o numerze t. W początkowej fazie uczenia $\delta(t)=\delta_{\max}$. Próg ten następnie zmniejsza się tak długo, aż nie osiągnięta zostanie minimalna wartość $\delta_{\min}$, zgodnie z następującym wzorem

$$\delta(t) = \max\left[\delta_{\max} \mathrm{e}^{-\frac{t}{\tau}}, \delta_{\min} \right] \tag{5.33}$$

τ jest stałą określającą szybkość zmian.

Rozmycie nowo utworzonej jednostki jest proporcjonalne do odległości do najbliższego istniejącego neuronu

$$\sigma = \kappa || \mathbf{D} - \mathbf{D}_{najbliższy} || \tag{5.34}$$

gdzie parametr κ jest czynnikiem określającym stopień nakładania się obszarów aktywacji dwóch rozpatrywanych węzłów.

Waga połączenia nowej jednostki z warstwą wyjściową ustalana jest na podstawie różnicy pomiędzy aktualnym żądanym wyjściem a wyjściem sieci

$$w = d - y(\mathbf{x}) \tag{5.35}$$

W przypadku, gdy węzeł nie jest dostawiany, następuje optymalizacja istniejących parametrów za pomocą reguły Widrowa-Hoffa

$$\Delta w_j = \alpha(d - y(\mathbf{x}))G_j \tag{5.36}$$

$$\Delta\theta = \alpha(d - y(\mathbf{x})) \tag{5.37}$$

W celu zmniejszenia błędu następuje również zmiana położeń istniejących węzłów zgodnie z następującą relacją

$$\Delta D_{jk} = 2\frac{\alpha}{\sigma_j}\left(x_k - D_{jk}\right)\left(d - y(\mathbf{x})\right) w_j G_j \tag{5.38}$$

6.5. Sieci RBF w zastosowaniach regułowych

Z uwagi na lokalny charakter funkcji gaussowskiej stosując logiką rozmytą można w naturalny sposób dokonywać interpretacji węzłów ukrytych w postaci reguł. Każdy z węzłów tworzy jedną regułę rozmytą postaci

Jeżeli (**x** *należy do* ***A***) *to y należy do B*

Gdzie ***A*** i *B* są zbiorami rozmytymi zdefiniowanymi w następujący sposób

$$A = \{(\mathbf{x}, \mu_A(\mathbf{x})) \mid \mathbf{x} \in X\} \tag{5.39}$$

gdzie $\mu_A(\mathbf{x}) : X \to [0,1]$ jest funkcją przynależności przyporządkowującą każdemu elementowi $\mathbf{x} \in X$ wartość rzeczywistą $\mu_A(\mathbf{x})$ z przedziału [0,1].

Niestety interpretacja reguł rozmytych jest znacznie trudniejsza od reguł logiki klasycznej, dlatego też powstały algorytmy np. RecBFN (Rectangular Basis Functions Network) pozwalające zastosować topologię oraz ogólny schemat uczenia sieci RBF, do problemów klasyfikacyjnych oraz do wyciągania klasycznych reguł logicznych.

6.5.1. RecBFN

Sieć RecBFN ma taką samą strukturę co sieć RBF tzn. węzły wejściowe, jedną warstwę ukrytą i warstwę wyjściową. Obydwie te sieci odróżniają zastosowane funkcje transferu oraz reguła propagacji sygnału.

Algorytm RecBFN tworzy zbiór hiperprostopadłościanów, reprezentujących jakąś określoną klasę, używając przy tym algorytmu uczenia opartego na algorytmie sieci RBF. Występowanie funkcji radialnych w sieciach RBF powoduje, że wiedza opisywana jest przez reguły rozmyte, dlatego też w celu utworzenia reguł klasycznych dokonano zmiany funkcji transferu z radialnej na funkcję postaci

$$R(\mathbf{x}) = \min_{1 \le i \le n} F(x_i, D_i, \sigma_i) \tag{5.40}$$

Funkcja *F* może mieć np. taką postać

$$F(x_i, D_i, \sigma_i) = \begin{cases} 1 : \mid x_i - D_i \mid \le \sigma_i \\ 0 : w\ przeciwnym\ razie \end{cases} \tag{5.41}$$

jak również

$$F(x_i; D_i, \sigma_i) = e^{\frac{(x_i - D_i)^2}{\sigma_i^2}} \tag{5.42}$$

W przypadku zastosowania funkcji (5.41) dowolny węzeł sieci, prototyp, który reprezentuje jakąś klasę *C*, ma bardzo prostą interpretację regułową postaci

$$Jeżeli\ \forall 1 \le i \le : x_i \in [D_i - \sigma_i, D_i + \sigma_i]\ to\ klasa\ C$$

Algorytm uczenia tej sieci oparty jest na algorytmie DDA (Dynamic Decay Adjustment) [3]. Jest to algorytm konstruktywistyczny bazujący na dwóch krokach:

1. Dostawianie – jeżeli nowy wzorzec nie jest pokrywany przez istniejący prototyp z poprawnej klasy, wprowadzana jest nowa jednostka. Pozycja nowej jednostki **D** jest taka sama jak aktualnego wzorca, natomiast rozmycie jest tak duże jak to tylko możliwe bez wprowadzania konfliktu z istniejącymi prototypami.

2. Zmniejszanie – jeżeli nowy wzorzec nie jest klasyfikowany poprawnie przez istniejący prototyp, rozmycia prototypu zostają zmniejszone tak, aby rozwiązać konflikt. Występują dwa cele zmniejszania rozmyć:
    - punkt konfliktowy musi leżeć poza obcinanym obszarem
    - powinno nastąpić jak najmniejsze zmniejszenie prostokąta, aby uniknąć powstawania małych reguł.

Pełny algorytm przedstawiony zostanie poniżej. Przyjęte zostały następujące oznaczenia: W_i^C - waga pomiędzy istniejącym *i*-tym węzłem w warstwie ukrytej a węzłem reprezentującym klasę *C*, p_i^C - i-ty prototyp klasy *C*, dwa zbiory rozmyć: zbiór wymiarów, w których rozmycia (prostokąty) zajmują całą przestrzeń K^∞, zbiór wymiarów, w których σ jest ograniczone K^e.

Na początku wszystkie wagi są zerowane. Następnie wszystkie wzorce treningowe prezentowane są na wejście sieci. Jeżeli nowy wzorzec klasyfikowany jest poprawnie waga najbliższego prototypu jest zwiększana, w przeciwnym przypadku wprowadzany jest nowy węzeł, którego położenie wyznaczone jest przez aktualny wzorzec wejściowy. W tym przypadku następuje również zmniejszanie rozmyć wszystkich prototypów reprezentujących nie zgodną klasę, których aktywacja jest zbyt wysoka.

ALGORYTM RecBFN
DLA WSZYTKICH PROTOTYPÓW p_i^C ***WYKONUJ***
$W_i^C = 0$
KONIEC
DLA WSZYSTKICH WZORCÓW CIĄGU TRENINGOWEGO WYKONUJ
JEŻELI $\exists p_i^C : R_i^C = 1$ ***TO***
$W_i^C = W_i^C + 1$
W PRZECIWNYM RAZIE
Dodaj nowy węzeł $\mathbf{p}_{m_C+1} = \mathbf{x}$
DLA WSZYSTKICH $1 \le i \le n$ ***WYKONUJ***
$\sigma_i = \infty$
DLA WSZYSTKICH $K \neq C, 1 \le j \le m_K$ ***WYKONUJ***
$p_{m_C+1}^C.Pomniejsz(p_j^K)$
$W_{m_C+1}^C = 1$
$m_C = m_C + 1$
KONIEC
DLA WSZYSTKICH $K \neq C, 1 \le j \le m_K$ ***WYKONUJ***
$p_j^K.Zmniejsz(\mathbf{x})$
KONIEC
KONIEC ALGORYTMU

W algorytmie zmniejszania rozmyć rozpatrywane są następujące przypadki:

1. Jeżeli istniejące skończone rozmycie może zostać zmniejszone bez zmniejszenia poniżej $\sigma_{\min}$, wybrana zostanie taka zmiana, która spowoduje najmniejsze zmniejszenie objętości hiperprostopadłościanu.
2. Jeżeli nie jest spełniony warunek 1, jedno z istniejących nieskończonych rozmyć zostanie zmniejszone lub jeżeli zmniejszenie to spowoduje powstanie rozmycia mniejszego niż $\sigma_{\min}$ jedno ze skończonych rozmyć zostanie zmniejszone poniżej wartości $\sigma_{\min}$.

ALGORYTM ZMNIEJSZ

$$\sigma_{\max,i} = \max\{|D_i - x_i| : \sigma_i \in K^{\infty}\}$$

$$\sigma_{\min,i} = \min\left\{|D_i - x_i| : \forall 1 \le l \le n,\ j \ne l : \frac{\sigma_j - |D_j - x_j|}{\sigma_j} \le \frac{\sigma_l - |D_l - x_l|}{\sigma_l}\right\}$$

$$\sigma_{best,k} = \min\left\{|D_k - x_k| : \forall 1 \le l \le n, k \ne l : \left(\frac{\sigma_k - |D_k - x_k|}{\sigma_k} \le \frac{\sigma_l - |D_l - x_l|}{\sigma_l}\right) \wedge \left(\sigma_l \ge \sigma_{i,\min}\right)\right\}$$

JEŻELI σ_{best} ***ISTNIEJE TO***

$$\sigma_k = \sigma_{best,k}$$

W PRZECIWNYM RAZIE JEŻELI $\sigma_{\max,i} \ge \sigma_{best,j}$

$$\sigma_i = \sigma_{\max,i}$$

ELSE

$$\sigma_j = \sigma_{\min,j}$$

KONIEC

KONIEC ALGORYTMU

7. Modyfikacje sieci MLP i ekstrakcja reguł logicznych.

Sieci neuronowe mogą w wielu przypadkach osiągać dobrą dokładność klasyfikacji, jednak z powodu dużej liczby parametrów, skomplikowanej architektury, trudno jest prześledzić działanie takiej sieci, dlatego też często nazywa się je są czarnymi skrzynkami. Uproszczenie architektury sieci MLP możliwe jest poprzez wprowadzenie różnych typów obszarów decyzyjnych realizowanych przez węzły ukryte. Zmiana obszarów decyzyjnych może nastąpić bez zmiany funkcji aktywacji. Modyfikacja sieci MLP umożliwiająca realizowanie różnych obszarów decyzyjnych przedstawiona jest w rozdziale 7.3.

Zrozumienie działania sieci jest możliwe poprzez zastosowanie zbioru reguł logicznych odpowiadających nauczonej sieci. Rozumowanie oparte o reguły logiczne jest znacznie bardziej przystępne dla człowieka, pozwala analizować otrzymane rezultaty jak również dostrzec istotne relacje między cechami, a co za tym idzie zwiększa zaufanie do systemu. Inną ważną przyczyną, dla której warto jest stosować systemy regułowe jest ich dokładność. Jak wynika z doświadczeń w wielu przypadkach systemy regułowe dają najlepsze rezultaty, zarówno na zbiorach treningowych jak i testowych, przy bardzo prostym zestawie reguł np. w danych medycznych.

Stworzono wiele systemów ekstrakcji reguł logicznych z danych. Wywodzą się one z teorii uczenia maszynowego (ML machine learning). Należą do nich drzewa decyzyjne, metody indukcyjne, jak również algorytmy oparte na zbiorach rozmytych czy też zbiorach przybliżonych. Mimo to reguły powstałe z sieci neuronowych w niektórych przypadkach okazują się być prostsze i dokładniejsze od tych powstałych z tak renomowanych algorytmów jak np. C4.5. Poza tym zastosowanie metod ekstrakcji reguł do sieci neuronowych pozwala lepiej zrozumieć, co tak naprawdę dzieje się w sieciach neuronowych.

Systemy regułowe powinny być jednak preferowane tylko w przypadkach, gdy zbiór reguł nie jest zbyt złożony oraz gdy ich dokładność jest wystarczająco duża. Zdarza się, że niektóre systemy regułowe produkują setki reguł, przez co nie są one bardziej użyteczne od systemu typu czarnej skrzynki.

7.1. Podstawowe metody ekstrakcji reguł z sieci neuronowych

Występują dwa typy algorytmów wyciągania reguł z sieci neuronowych:

- metody globalne
- metody lokalne.

Metody globalne polegają na analizie wyjścia całej sieci dla różnych wejść. Jedną z takich metod jest VIA (Validity Interval Analysis) rozwijana przez Thruna [62]. Algorytm ten polega na utworzeniu dla każdego neuronu (lub też podzbioru neuronów) interwałów zasięgu aktywacji takich, że aktywacja sieci musi leżeć wewnątrz wyznaczonych interwałów.

Załóżmy, że mamy wyznaczony zbiór interwałów *I*. Wektor aktywacji nazywany jest dopuszczalnym przy zadanym zbiorze *I* wtedy, gdy wszystkie aktywacje leżą w *I*. Zbiór interwałów jest zgodny, jeżeli istnieje dopuszczalny wektor aktywacji.

Niech będzie dany początkowy zbiór interwałów *I*. Zbiór ten w sposób iteracyjny jest oczyszczany poprzez techniki programowania liniowego w taki sposób, aby wykluczyć te aktywacje, które są nie dopuszczalne.

Reguły uzyskiwane w tej metodzie mają postać warunków, które dane są przez zbiór interwałów dla danego wektora wejściowego, natomiast konkluzja reprezentuje pojedynczą kategorię

$$Jeżeli\ x \in hiperszecianu\ I\ to\ klasa\ C$$

Jedną z wad tego algorytmu jest to, że ma on tendencje do produkowania dużej liczby reguł.

Metody lokalne dokonują ekstrakcji reguł na podstawie analizy fragmentu sieci, zazwyczaj pojedynczego węzła ukrytego. Zestaw reguł dla całej sieci powstaje poprzez kombinowanie reguł otrzymanych z fragmentów sieci. W sieciach takich stosuje się sigmoidalne funkcje transferu (w ograniczeniu logicznym funkcje skokowe), lub też funkcje zlokalizowane. Użycie funkcji skokowej powoduje, że wyjście każdego neuronu jest binarne. Ponieważ funkcja sigmoidalna jest funkcją monotoniczną a jej wartości są z przedziału [0,1] wystarczy znać tylko znak wagi w sieci, aby wyznaczyć jej wkład do aktywacji danej jednostki. Występuje więc 2^n (n liczba cech wejściowych) możliwych kombinacji. Reguły dla całej sieci są tworzone z reguł powstałych dla poszczególnych węzłów. Metody tego typu mają problemy z eksponencjalnym wzrostem liczby możliwych reguł. Jednym z rozwiązań są reguły *M*-z-*N* zaproponowane przez Towell i Shavlik [65] np. reguła postaci 2 z $\{a,b,c\}$ daje się przedstawić w postaci reguły z 3 przesłankami $\{a \wedge b\} \vee \{a \wedge c\} \vee \{b \wedge c\}$, lub 3 reguł posiadających 1 przesłankę. Reguły *M*-z-*N* doskonale nadają się do sieci neuronowych, gdyż każdy z węzłów reprezentuje jedną taką regułę. Uniknięcie eksplozji kombinatorycznej jest również możliwe przez tworzenie grup połączeń posiadających podobne wartości wag. Wartości wag w takich grupach zastępowane są wartościami średnimi. Grupy, które nie mają wpływu na aktywację zostają odrzucone, natomiast progi, zoptymalizowane przy zamrożonych wagach. Taka uproszczona sieć jest łatwiejsza do analizy gdyż ma mniejszą liczbę niezależnych wejść.

Inną metodą uproszczenia sieci neuronowej jest metoda zaproponowana przez Setiono i Liu [57] oparta na dodaniu do funkcji kosztu członu regularyzacyjnego powodującego obcinanie małych wag. Po uproszczeniu wykonywana jest dyskretyzacja sieci za pomocą klasteryzacji wartości aktywacji, dla jednostek ukrytych, otrzymanych po zaprezentowaniu całego zbioru treningowego. Metoda ta nadaje się niestety tylko do małych sieci. Inną metodą opartą na podobnej zasadzie jest SR (Succesive Regularisation) [30] z członem regularyzacyjnym będącym sumą wartości bezwzględnych wag. Tylko wagi mniejsze od pewnego zadanego progu są uwzględniane w członie regularyzacyjnym (tzw. selektywne zapominanie). W trakcie uczenia wymusza się, aby węzły ukryte zawsze były w pełni aktywne lub zupełnie nieaktywne. Po nauczeniu zostawia się tylko szkielet sieci a następnie dokonywana jest ekstrakcja dominujących reguł. Szkieletowa postać sieci zostaje zamrożona, następnie poprzez zmniejszenie parametru regularyzacyjnego dopuszczona zostaje możliwość tworzenia małych wag. Po nauczeniu z mniejszą wartością parametru regularyzacyjnego otrzymuje się bardziej złożoną sieć, a poprzez analizę nowych połączeń uzyskuje się bardziej szczegółowe reguły. Inna prosta metoda należąca do tej grupy metod została zaprezentowana przez Geczy i Usui [25]. Wagi w sieci MLP z jedną warstwą ukrytą po nauczeniu mapowane są do wartości –1,0,+1. Powoduje to znaczne uproszczenie etapu poszukiwania reguł. Podobna idea przyświeca metodzie MLP2LN [16] opisanej szczegółowo w rozdziale 7.2.

Algorytm DEDEC [63] wyciąga reguły poprzez znajdowanie minimalnego zestawu cech pozwalającego oddzielić, z punktu widzenia sieci neuronowej, dany wzorzec od pozostałych znajdujących się w zbiorze treningowym. Tworzony jest nowy zbiór treningowy przez zastępowanie oryginalnych przypadków całymi grupami, wyznaczonymi na podstawie analizy skupisk. Dokonywany jest ranking wejść, uporządkowany ze względu na ważność, poprzez

przegląd wpływu wag wejściowych na wyjście sieci. Reguły tworzone są za pomocą metod szukania, ale w przesłankach reguł używane są tylko te cechy, które podczas dokonywania rankingu uznane zostały jako ważne.

Zamiast szukać bezpośrednio logicznego opisu danych można próbować opisać za pomocą reguł logicznych działanie dowolnego klasyfikatora. Jeśli udało się nam stworzyć dobry klasyfikator (np. sieć neuronową, nazywany „wyrocznią"), można go wykorzystać do odpowiedzi na wiele pytań. Jednym z takich algorytmów jest REAL (Rule Extraction As Learning), który jest ogólną techniką inkrementacyjnego generowania reguł stworzoną przez Cravena i Shavlik [10]. Jeżeli nowy przypadek nie jest poprawnie klasyfikowany przez istniejący zbiór reguł, tworzona jest nowa reguła bazująca na tym przypadku. Nowa reguła jest zainicjowana w ten sposób, że wszystkie cechy danego przypadku są przesłankami tej reguły. Następnie odrzucane są poszczególne przesłanki i sprawdzane jest czy reguła jest zgodna z siecią. Jeżeli tak, to przesłanka jest odrzucana, jeżeli nie pozostaje w regule. Na podobnej zasadzie działa algorytm RULENEG [1]. Jako przesłanki do nowej reguły dodawane są te wejścia neuronu, które mają wpływ na klasę reprezentowaną przez dany wektor. Wpływ jest wyznaczany przez negowanie kolejnych wartości wejściowych i sprawdzanie (za pomocą sieci neuronowej) czy nastąpiła zmiana klasyfikacji. Jednym z bardzo efektywnych systemów działających w oparciu o wyrocznię jest TREPAN stworzony przez Cravena i Shavlika [11], który generuje drzewo decyzji na podstawie analizy odpowiedzi sieci neuronowej dla przedstawionych jej próbek danych. Podstawową zaletą takiego rozwiązania jest uniezależnienie systemu ekstrakcji reguł od zbioru danych. Nowe próbki mogą być generowane tak, by zapewnić wystarczającą liczbę przypadków w tych obszarach, w których same dane dostarczają niewiele informacji. Można w ten sposób poszukiwać logicznego opisu działania każdego systemu klasyfikującego (np. korzystającego z metod opartych na podobieństwie, metod statystycznych itp.). Istotną wadą takiego podejścia jest fakt, że tak powstałe reguły mogą znacznie odbiegać od wyjściowych danych, ponieważ nakładają się tutaj dwa różne błędy uczenia - pierwszy podczas uczenia badanego klasyfikatora, a drugi podczas próby opisu jego działania. Z tego powodu bardziej uzasadnionym podejściem do celu ekstrakcji reguł logicznych z surowych danych wydaje się być modyfikowanie algorytmów uczenia systemów sztucznej inteligencji w taki sposób, by bezpośrednio po nauczeniu systemu móc z łatwością opisać jego działanie przez zbiór reguł logicznych.

7.2. MLP2LN

MLP2LN jest to algorytm zamiany sieci MLP na sieć realizującą funkcje logiczne (LN - Logical Network), co pozwala na wyciąganie reguł logicznych, stąd MLP2LN. Polega on na gładkim przejściu od sieci MLP do sieci, w której w prawie wszystkie parametry adaptacyjne (wagi) posiadają wartości +1, –1,0, natomiast progi mają dowolną wartość. Funkcję realizowaną przez tak przygotowaną sieć można w bardzo łatwy sposób przedstawić w postaci reguł logicznych.

W tym podrozdziale przedstawiona została metodologia ekstrakcji reguł logicznych za pomocą sieci neuronowej MLP2LN. W pierwszej części występuje opis tworzenia początkowych zmiennych lingwistycznych. W części drugiej zaprezentowana została architektura sieci MLP2LN. Trzecia część zawiera funkcję kosztu stosowaną w sieci MLP2LN wraz z różnymi możliwymi odmianami tej funkcji. W części czwartej opisany został algorytm uczenia. W piątej interpretacja węzłów ukrytych oraz proces tworzenia reguł. W szóstej wprowadzone zostały jednostki L pozwalające w sposób automatyczny dokonywać kwantyzacji cech.

7.2.1. Zmienne lingwistyczne

Reguły logiczne wymagają na wejściu zmiennych o wartościach symbolicznych (zmiennych lingwistycznych). Dlatego też zanim rozpoczęty zostanie proces wyciągania reguł należy wstępnie przygotować dane ze zbioru treningowego tzn. dokonać ich kwantyzacji. W przypadku, gdy dana cecha przyjmuje wartości ciągłe, w fazie wstępnej obróbki trzeba podzielić ją na kilka rozdzielnych zbiorów i wprowadzić nowe zmienne logiczne, nazywane zmiennymi lingwistycznymi. Określa się je za pomocą funkcji logicznych, czyli predykatów działających na pojedynczych zmiennych rzeczywistych. Na przykład zmienna *rozmiar* będzie miała wartość *mały*, jeżeli zmienna ciągła *x* znajdzie się w określonym przedziale, $x \in [a,b]$, może też mieć wartości *duży* czy *średni*. *Mały*(*x*) jest więc predykatem przyjmującym wartości 1 (prawda) dla *x* z określonego przedziału lub dla *x* będącego określonym podzbiorem. Wówczas reguła logiczna przyjmuje następującą postać:

Jeżeli (s_k=*mały* lub s_k=¬*duży*) *To Klasa1*

gdzie ¬ oznacza „nieprawda że". Każda zmienna *s* może mieć dwie lub więcej wartości. Jeden ze sposobów reprezentacji numerycznej zmiennych lingwistycznych jest wektor $V_{s_1} = (+1,-1,-1,...)$ dla pierwszej wartości tej zmiennej, [ ] dla drugiej wartości itp. dla pozostałych. Wartość +1 może wystąpić tylko w jednym miejscu wektora *V*, pozycja ta określa kodowaną wartość lingwistyczną. Zakodowanie zmiennej w tej postaci umożliwia wyprowadzenie reguł zawierających zarówno potwierdzenie np. [ ] (dzięki występowaniu wartości +1) oraz zaprzeczenia (wartość -1). Jeśli wektor nie posiada wartości dla danej cechy (czyli jest to cecha brakująca), można wstawić 0 na każdej pozycji tej cechy. Zero w tym wypadku można zinterpretować jako wartość, która nie ma wpływu na wzbudzenia węzła. Kwantyzacja implementowana jest poprzez węzły w warstwie wejściowej, co powoduje, że zwiększa się liczba węzłów w porównaniu do sieci, w której mogą wystąpić zmienne ciągłe. Np. jeżeli mamy 4 cechy i każda z nich kodowana jest za pomocą trzech zmiennych lingwistycznych, to powstaje sieć z 12 węzłami wejściowymi. Ta „nadmiarowość" węzłów wejściowych (a więc i parametrów adaptacyjnych) umożliwia wyprowadzanie bardzo prostych reguł logicznych, gdyż większość z tych wejść okazuje się w trakcie uczenia niepotrzebna. Wyjście sieci natomiast kodowane jest przez liczby 0 i 1. Wartość 1 na wyjściu oznacza, że wektor jest z tej klasy natomiast dla wszystkich pozostałych klas musi być wartość 0.

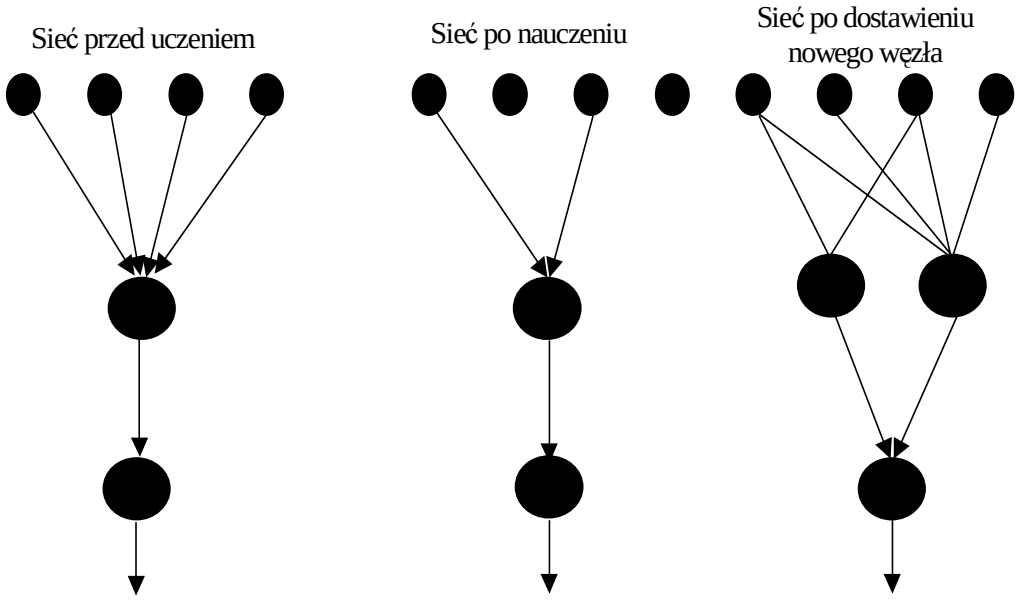
7.2.2. Architektura sieci

MLP2LN składa się z trzech głównych warstw: wejściowej, ukrytej (nazywana również warstwą *R*, regułową), wyjściowej. Występuje tylko jedna warstwa *R*. Teoretycznie możliwe jest wprowadzenie kilku warstw *R*. Prowadzi to jednak do tworzenia meta reguł, tzn. reguł, których przesłanki są regułami. Tworzenie meta reguł może być użyteczne tylko wtedy, gdy reguły nie są regułami rozłącznymi tymczasem w sieci MLP2LN tak nie jest. Między warstwą *R* a warstwą wejściową mogą wystąpić jednostki, których zadaniem jest dokonanie kwantyzacji cech np. jednostki L 7.2.6 składające się z 2 warstw. W najprostszym przypadku, gdy wszystkie cechy w zbiorze treningowym poddane zostały kwantyzacji, sieć MLP2LN ma tylko 3 warstwy. Liczba węzłów w warstwie wyjściowej równa jest liczbie klas w zbiorze treningowym, natomiast w warstwie wejściowej liczbie zmiennych lingwistycznych. Można wyróżnić dwie metody pozwalające określić liczbę węzłów w warstwie ukrytej *R*. Pierwsza z niech okre-

ślana jako C-MLP2LN startuje z jednego neuronu dla każdej z klas (wersja konstruktywistyczna), druga startuje z dużej liczby węzłów i w trakcie uczenia na skutek mocnego wymuszania małych wartości wag liczba parametrów adaptacyjnych ulega redukcji. W dalszej części bardziej szczegółowo omówiona zostanie wersja C-MLP2LN.

Początkowa liczba węzłów w warstwie ukrytej *R* równa jest na ogół liczbie węzłów w warstwie wyjściowej. Węzły w warstwie ukrytej połączone są ze wszystkimi węzłami z warstwy wejściowej i tylko jednym węzłem w warstwie wyjściowej. Podczas procesu uczenia zmianie ulegają tylko wagi pomiędzy warstwą wyjściową i ukrytą. Pozostałe wagi mają wartość ustaloną na +1 lub –1, co jest wynikiem stosowania zmiennych lingwistycznych. Analizując wagi i próg w danym neuronie ukrytym otrzymujemy reguły odnoszące się do klasy, z którą ten neuron jest połączony (połączenie z określonym węzłem wyjściowym). Jeśli waga połączenia jest +1, to otrzymujemy reguły dla danej klasy, jeśli natomiast waga jest –1, to wyjątki, czyli reguły opisujące przypadki błędnie klasyfikowane przez istniejące reguły dla tej klasy. Węzły w warstwie wyjściowej dokonują jedynie sumowania aktywacji węzłów z warstwy ukrytej. W przypadku, gdy dwa węzły klasyfikują ten sam wektor traktowane jest to przez sieć jako błąd, gdyż na wyjściu na skutek sumowania pojawia się wartość większa od 1. Otrzymujemy dzięki temu z różnych węzłów zawsze reguły, które są rozłączne, czyli nie klasyfikują tych samych wektorów.

Proces uczenia odbywa się dla każdego wyjścia (klasy) niezależnie. Na początku istnieje tylko jeden neuron ukryty (dla danej klasy), który trenowany jest na wszystkich wektorach z ciągu treningowego. Do zmiany wag stosowana jest standardowa procedura wstecznej propagacji z momentem. Po zakończeniu uczenia dostawiany jest nowy węzeł do warstwy ukrytej, który jest łączony z tą samą klasą. Poprzedni węzeł natomiast jest zamrażany tzn. wagi tego węzła podczas dalszego uczenia nie będą się zmieniały. Dzięki temu wektory wejściowe, które są poprawnie klasyfikowane przez węzeł zamrożony nie dają już wkładu do funkcji błędu. Sieć uczona jest ponownie i w razie konieczności dostawiany jest następny neuron. W przypadku, gdy istniejące zamrożone węzły popełniają błędy dostawiany jest węzeł z wagą -1 i szukane są wyjątki dla istniejących węzłów, czyli następuje próba znalezienia reguł, która opiszą wektory dotychczas błędnie klasyfikowane. Cała procedura powtarzana jest tak długo, aż uzyskany zostanie wystarczająco mały błąd lub też, aż reguły, które powstają podczas analizy ostatnio nauczonego węzła, stają się zbyt szczegółowe, lub jest ich zbyt dużo.



Reguły powstające z pierwszych węzłów są najbardziej ogólne. Kolejne neurony dają coraz bardziej szczegółowe reguły, aż wreszcie powstają reguły opisujące pojedyncze wektory. Oczywiście reguły opisujące niewielką liczbę wektorów są zwykle pomijane, ponieważ reprezentują szum w danych lub bardzo szczegółowe przypadki. Może się jednak zdarzyć, że takie reguły będą interesujące dla eksperta w danej dziedzinie, gdyż reprezentują rzadkie przypadki, nie należy więc ich odrzucać automatycznie. Dlatego też można powiedzieć, że reguły otrzymywane są w uporządkowanej kolejności, od najbardziej ogólnych do coraz bardziej szczegółowych. Proces uczenia jest bardzo szybki, ponieważ w danej chwili uczony jest tylko jeden (dla każdej klasy) węzeł sieci. W trakcie uczenia sieć jest konstruowana poprzez dodawanie kolejnych węzłów.

7.2.3. Funkcja kosztu

Interpretacja węzłów w sieci MLP jest w ogólności bardzo trudna. Można ją znacznie ułatwić używając sigmoid z bardzo dużym skosem, oraz wymuszając wartości wag, pomiędzy warstwą wejściową i ukrytą, na 0,+1,-1. Wartość 0 oznacza, że zmienna wejściowa połączona tą wagą jest nieistotna, +1 oznacza, że dana cecha musi wystąpić, a -1 nie może wystąpić. Można to osiągnąć poprzez modyfikacje funkcji błędu stosowanej dla algorytmu wstecznej propagacji.

$$E(w) = \frac{1}{2}\sum_p \sum_i (y(\mathbf{x}_i, \mathbf{w}) - d_i^p)^2 + \frac{\lambda_1}{2}\sum_{i>j} w_{ij}^2 + \frac{\lambda_2}{2}\sum_{i>j} w_{ij}^2 (w_{ij}+1)^2 (w_{ij}-1)^2 \qquad (5.43)$$

wówczas gradient dodatkowych członów ma postać

$$\lambda_1 w_{ij} + \lambda_2 w_{ij}(w_{ij}^2 - 1)(3w_{ij}^2 - 1) \qquad (5.44)$$

Pierwszy człon równania (5.43) mierzy różnicę między żądanym wyjściem dla wektorów z ciągu treningowego, a wyjściem otrzymanym z sieci. Drugi człon wymusza podczas uczenia małe wartości wag, przez co prowadzi do eliminacji cech zbędnych, człon trzeci natomiast wymusza wartości wag -1,0,+1 umożliwiając późniejszą logiczną interpretację sieci. Za pomocą parametrów λ_1, λ_2 można zwiększać lub też zmniejszać dominacje członu drugiego i trzeciego. Ustalenie dominacji któregoś z członów wyznacza granicę pomiędzy prostotą a dokładnością reguł otrzymanych z sieci. Jeżeli chcemy uzyskać bardzo prostą sieć (proste reguły), które dają tylko przybliżony opis danych, pierwszy człon powinien być tak duży, jak to tylko jest możliwe, przy akceptowalnym jeszcze błędzie.

Człon wymuszający wartości 0,+1, −1 (6-tego stopnia) może zostać zastąpiony przez człon o innej postaci:

$$|w_{ij}||w_{ij}^2 - 1| \text{ człon sześcienny}$$
$$|w_{ij}| + |w_{ij}^2 - 1| \text{ człon kwadratowy}$$

lub też przez człon Weigenda (4.39). Zastosowanie członu Weigenda umożliwia uzyskanie sieci z wagami o dowolnych wartościach. Wagi, które w wyniku działania członu wy-

muszającego zera są małe, zostają przez człon Weigenda zepchnięte do zera, pozostałe wagi natomiast mogą osiągać duże wartości. W tym wypadku uzyskanie wag całkowitych możliwe jest poprzez zastosowanie dodatkowego członu postaci:

$$\sin(\frac{3}{4}\pi w_{ij}) \tag{5.45}$$

Człon ten uaktywniany jest po nauczeniu za pomocą członu Wigenda, który zostaje wtedy wyzerowany. Może również być zastosowany zamiast członu wymuszającego 1,0, –1. Jednak z uwagi na to, że na początku wymuszane są małe wartości wag, to w wyniku zadziałania (5.45) w końcowym etapie wagi zazwyczaj osiągają wartości +1, –1,0 i raczej rzadko większe. W niektórych przypadkach wymuszanie większych wartości wag umożliwia szybsze znalezienie lokalnego minimum.

Wprowadzanie członów regularyzacyjnych występujących w funkcji kosztu (5.43) ma swoja interpretacje z Bayesowskiego punktu widzenia [30], [38]. Funkcja kosztu zawiera naszą wiedzę a priori na temat rozkładu prawdopodobieństwa $P(\mathbf{w}|M)$ wag naszego modelu M. W problemach klasyfikacji, gdy wymagane są reguły logiczne, prawdopodobieństwo a priori wartości wag powinno zawierać nie tylko małe wagi, ale również duże dodatnie i ujemne rozmieszczone w pobliżu ±1 np.

$$P(\mathbf{w} \mid M) = Z(\alpha)^{-1} e^{\alpha E(\mathbf{w}|M)} \propto \prod_{ij} e^{-\alpha_1 w_{ij}^2} \prod_{ij} e^{-\alpha_2 |w_{ij}^2 - 1|} \tag{5.46}$$

gdzie parametry α_1, α_2 odgrywają tą samą rolę co parametry λ we wzorze (5.43).W przypadku zastosowania członu regularyzacyjnego zawierającego kwadrat wag (człon 2 we wzorze (5.43)), rozkład a priori jest rozkładem gaussowskim. Natomiast dla członu postaci

$$\sum_{i,j} | w_{ij} | \tag{5.47}$$

rozkładem a priori jest rozkład eksponencjalny. Wiedza początkowa o problemie może również być wprowadzona bezpośrednio do struktury sieci, poprzez zdefiniowanie warunków początkowych modyfikowanych podczas uczenia. Takim warunkiem początkowym mogą być reguły, które są niezbyt dokładne. Po zbudowaniu odpowiedniej struktury sieci odpowiadającej tym regułom poprzez uczenie można próbować zwiększyć dokładność reguł początkowych.

Chociaż te dodatkowe człony w funkcji błędu nie wystarczą by zamienić MLP dokładnie w sieć logiczną, to ułatwiają logiczną interpretacje końcowej sieci.

7.2.4. Proces uczenia

Algorytm uczenia dla pojedynczej klasy wygląda w następujący sposób:

1. Utwórz węzeł ukryty.
2. Ucz sieć za pomocą algorytmu propagacji wstecznej z regularyzacją. Użyj następujących wartości parametrów regularyzacyjnych: $\lambda_1=10^{-5}$, $\lambda_2=0$, skos $s=1$.
3. Jeżeli zbieżność jest słaba (np. po ustalonej liczbie epok błąd zmniejszył się mniej niż $1/n$, gdzie n jest liczbą wektorów zbioru treningowego) spróbuj trenować równocześnie 2 jednostki ukryte; w pewnych przypadkach trenowanie

więcej niż jednego neuronu równocześnie może znacznie przyspieszyć proces uczenia.

a) Ucz tak długo jak długo błąd się zmniejsza; zwiększ $\lambda_1=10\lambda_1$, zmień skos funkcji sigmoidalnych $s=s+1$ i ucz dalej; powtarzaj te kroki tak długo aż nastąpi gwałtowny wzrost wartości błędu (typową wartością jest wzrost 5 krotny).
b) Zmniejsz λ_1 o niewielką wartość tak aby uzyskać poprzednią (przed gwałtownym wzrostem) wartość błędu i ucz tak długo jak długo błąd maleje.
c) Użyj następujących wartości parametrów regularyzacyjnych: $\lambda_2=\lambda_1$, $\lambda_1=0$, trenuj sieć stopniowo zwiększając parametr λ_2 tak długo aż wagi istniejące w sieci osiągną wartości 0 ± 0.05 lub $\pm1\pm0.05$.
d) Ustaw skos funkcji sigmoidalnych na bardzo dużą wartość $s\approx10000$, wartości wag na wartości całkowite 0, ±1.

4. Przeanalizuj wagi i próg (progi), sprawdzając kombinacje zmiennych lingwistycznych powodujące aktywację neuronu (neuronów). Analiza ta pozwala utworzyć reguły logiczne (opis znajduje się w części 7.2.5).
5. Zamroź wagi istniejących węzłów. Jest to równoważne z uczeniem tylko nowych węzłów. Wagi zamrożone w dalszym etapie nie będą zmieniane.
6. Dodaj nowy neuron
7. Powtarzaj całą procedurę dopóty dopóki wszystkie dane nie zostaną poprawnie sklasyfikowane lub też liczba reguł, które się uzyskuje nie rośnie gwałtownie.
8. Wykonaj opisaną procedurę dla pozostałych klas.

Na początku procesu uczenia parametr $\lambda_2=0$ natomiast λ_1 jest mały (ma wartość 0.00001). Z takimi parametrami sieć uczona jest tak długo, jak długo maleje błąd. Następnie zwiększana jest wartość parametru λ_1 (np. 10 λ_1) i ponownie uczona jest sieć. Z reguły po zwiększeniu parametru λ_1 następuje wzrost błędu SSE; można zmniejszyć go poprzez zwiększenie skosów. Procedura ta powtarzana jest tak długo, aż większość wag będzie miała wartość bliską zeru lub też nastąpi bardzo duży skok błędu. W tym momencie człon odpowiedzialny za wymuszenie małych wartości wag przestaje być ważny i uaktywniany zostaje człon trzeci. Wartość parametru λ_2 jest równa lub też trochę większa od ostatniej wartości parametru λ_1. Jednocześnie zwiększane jest nachylenie sigmoidy węzłów w sieci. W celu jeszcze większej redukcji wag parametr λ_1 może pozostać niezerowy jednocześnie z parametrem λ_2. Jego wartość powinna być mniejsza niż λ_2 i przy dalszym zwiększaniu λ_2 przestaje być istotna i może zostać wyzerowana. W przypadku niektórych danych trzeba próbować różnych sposobów zmian parametrów po to, by uzyskać najprostsze reguły. Proces uczenia kontynuowany jest przy jednoczesnym zwiększaniu wartości parametru λ_2 oraz nachylenia sigmoid. Wraz ze wzrostem wartości parametru λ_2 zmniejszana jest wartość parametru uczenia, dzięki czemu człon trzeci staje się coraz ważniejszy, a co za tym idzie wagi mają wartości coraz bliższe +1,0, –1. Parametr λ_2 nie powinien przekraczać wartości 1. Jeśli osiągnął już swoją maksymalną wartość, natomiast parametr uczenia jest wciąż duży tzn. >0.00001, to wartości λ_2 nie jest już zmieniana a następuje jedynie zwiększanie skosu i zmniejszenie parametru uczenia, aż osiągnięta zostanie wartość minimalna (np.0). W końcowym etapie zerowane są wszystkie parametry i zwiększane jest nachylenie sigmoid do bardzo dużych wartości (10000), przez co uzyskuje się ostre granice decyzyjne. Początkowe wymuszenie małych wartości wag umożliwia w późniejszym etapie wyzerowanie tych wag zupełnie, pozostałe wagi natomiast będą zwiększały (zmniejszały) swoją wartość do +1 (–1) dzięki trzeciemu członowi. Może się zdarzyć, że na skutek zbyt mocnego wymuszenia wag o małych wartościach w pierwszej fazie, w końcowym etapie uzyskuje się węzeł, który posiada wszystkie wagi zerowe. W takim przypadku trzeba węzeł zainicjować od początku i zmniejszyć końcową wartość parametru λ_1.

Pomimo tego, że z węzła otrzymuje się proste reguły dobrze jest spróbować ponownie nauczyć sieć, ale z jeszcze większym wymuszeniem początkowych zer. Często zdarza się, że taka procedura prowadzi do jeszcze prostszej postaci sieci. Liczba reguł, która zostanie utworzona z danego węzła bardzo mocno zależy od liczby niezerowych wag, dlatego też etap pierwszy (wymuszanie małych wag) jest bardzo istotny. Cała procedura wymuszania wartości wag w sieci dotyczy tylko i wyłącznie wag, wszystkie progi w sieci mogą przyjmować dowolne wartości.

7.2.5. Interpretacja węzłów ukrytych

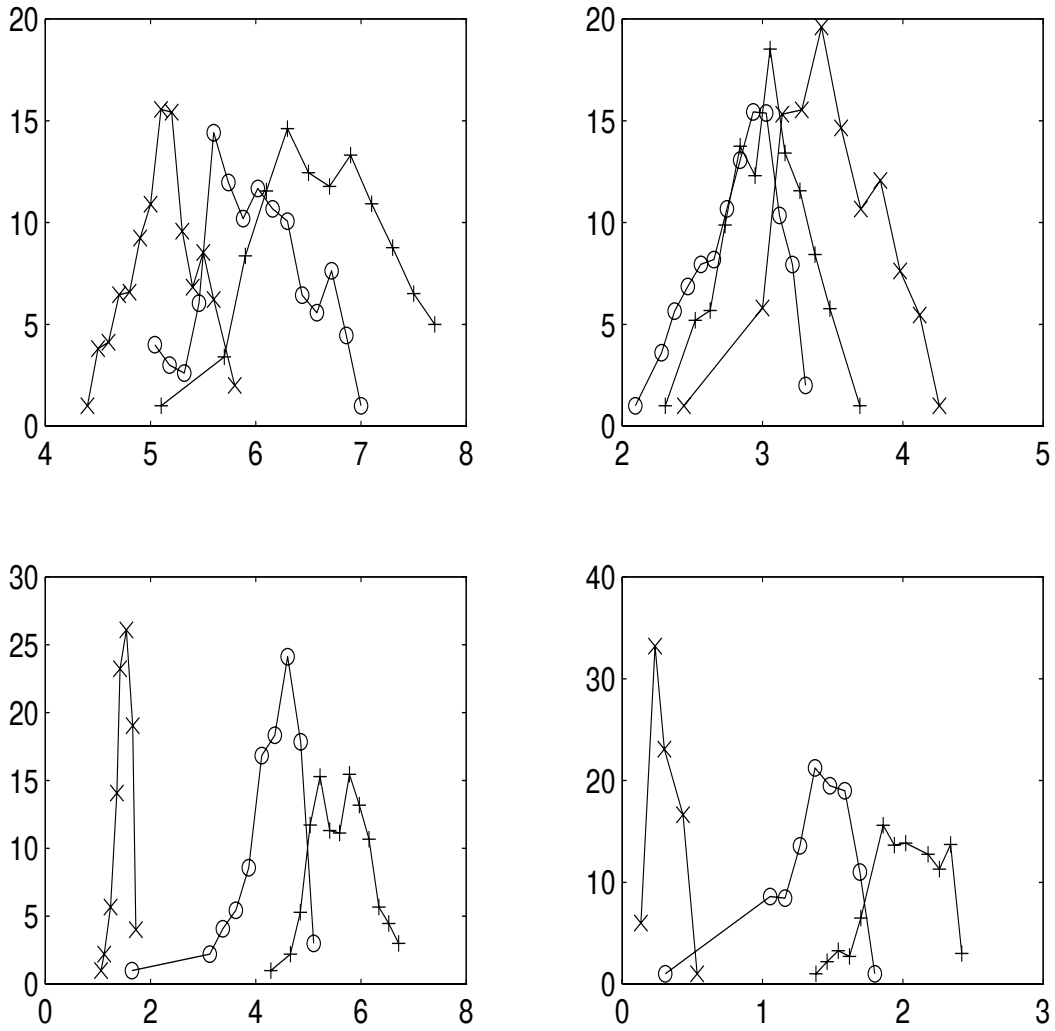
Wszystkie sygnały wejściowe oraz wagi pomiędzy warstwą wejściową a wyjściową mają wartości +1, –1,0, dlatego też sygnał wpływający do węzła ukrytego musi mieć wartość całkowitą. Ponieważ sigmoidy w węzłach mają bardzo duży skos (są bardzo ostre, skos jest 1000) to funkcja aktywacji ma wartość +1,0. Na podstawie analizy sygnału wpływającego oraz progu sigmoidy możemy określić w jakich przypadkach węzeł może się wzbudzić. Wzbudzenie może nastąpić tylko wtedy, gdy wartość sygnału wpływającego przekroczy wartość progu, ponieważ funkcja aktywacji ma postać

$$f(x) = \frac{1}{1 + e^{-s(\mathbf{wx}+\theta)}}$$

to $f(x)=1$ gdy $e^{-s(\mathbf{wx}+\theta)}$ jest równe zero, a więc gdy $\mathbf{wx}+\theta>0$ przy założeniu, że próg θ jest bardzo duży. Stąd żeby utworzyć reguły wystarczy zbadać, kiedy $\mathbf{wx}>-\theta$.

Poniżej przedstawiony zostanie przykład zastosowania sieci MLP2LN do wyciągania reguł logicznych na danych *iris*. Dane te zawierają 150 wektorów równomiernie podzielonych na trzy klasy: iris setosa, iris versicolor, iris virginica. Każdy wektor opisywany jest przez cztery cechy, podane w centymetrach: długość liścia x_1, szerokość liścia x_2, długość płatka x_3, szerokość płatka x_4.

Wartości oraz liczba zmiennych lingwistycznych została wyznaczona poprzez analizę histogramów dla każdej z cech niezależnie. Przedziały dla poszczególnych zmiennych lingwistycznych otrzymywane są na podstawie cięcia histogramów na obszary, w których występują wektory charakterystyczne dla danej klasy.



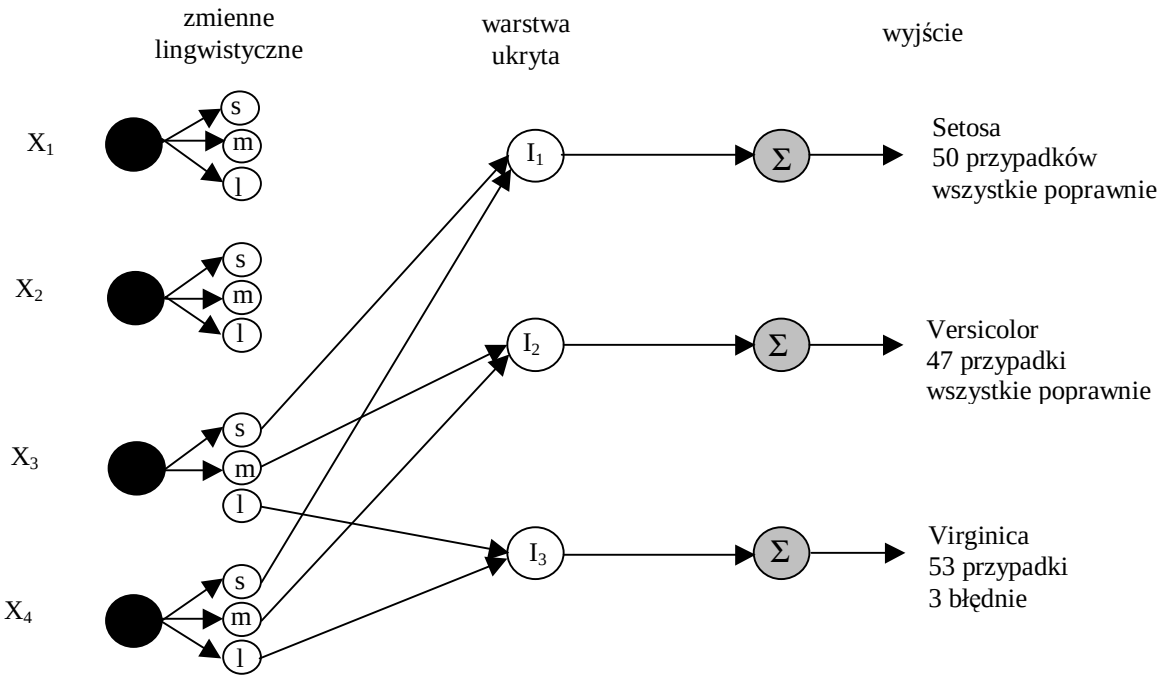
Rys 7:1 Histogramy dla poszczególnych wymiarów danych iris

Otrzymane zmienne lingwistyczne przedstawione są w poniższej tabeli

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

W wyniku kwantyzacji część wektorów treningowych ma tą samą reprezentację. Wektory o tej samej postaci, ale należące do innych klas, z uwagi na niejednoznaczność zostają usunięte z ciągu treningowego. Występuje 6 takich wektorów (zakodowane jako (m,m,l,l) oraz (m,l,m,l)) z klas *iris virginica* (3 wektory) i *iris versicolor* (3 wektory). Z ciągu treningowego usunięte zostają 3 wektory z klasy *iris versicolor*. W zbiorze treningowym pozostaje 147 wektorów, co w najlepszym przypadku daje 98% poprawnej klasyfikacji na zbiorze 150 wektorów.

Do nauczenia danych wystarczy sieć z 12 węzłami wejściowymi i z jednym węzłem ukrytym przypadającym na każdą z klas. Po nauczeniu otrzymuje się sieć postaci



Rys 7:2 Sieć MLP2LN po nauczeniu danych iris

Po nauczeniu powstaja sieć z następującymi parametrami

- iris setosa: (0,0,0;0,0,0;+1,0,0;+1,0,0), θ=1
- iris versicolor: (0,0,0;0,0,0;0,+1, –1;0,+1, –1), θ=3
- iris virginica: (0,0,0;0,0,0; –1, –1,+1; –1, –1,+1), θ=1

Ponieważ dla klasy iris setosa niezerowe wagi występują tylko dla cechy x_3 oraz x_4 (+1,0,0), to powstała reguła logiczna przy progu θ=1 jest bardzo prosta i ma postać

Jeżeli $(x_3=s \wedge x_4=s)$ to *iris setosa*

Dla klasy iris versicolor z uwagi na występowanie dwóch wag na raz obie cechy x_3 i x_4 dają wkład +2 lub –2 czyli razem 4,0 lub –4. Aby przekroczyć próg θ=3 obie cechy muszą dać dodatni wkład, stąd reguła ma postać

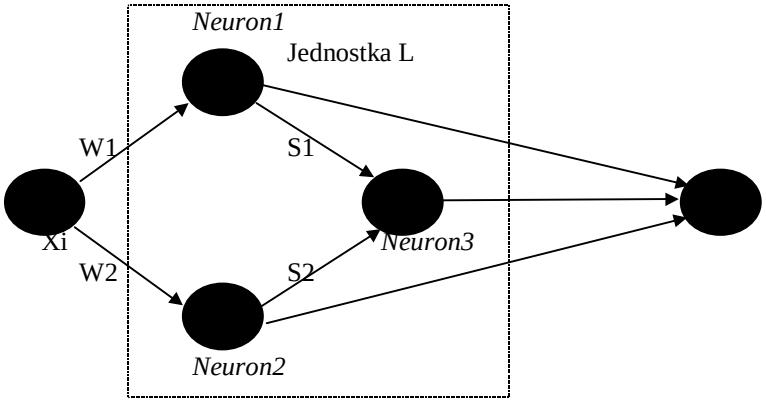
Jeżeli $(x_3=m \wedge x_4=m)$ to *iris versicolor*

Dla klasy iris virginica każda z cech daje wkład +3 lub –1 stąd, aby przekroczyć próg θ=1 wystarczy wkład z jednej cechy. Reguła dla iris virginica ma postać

Jeżeli $(x_3=l \vee x_4=l)$ to *iris virginica*

7.2.6. Jednostki lingwistyczne

Algorytm opisany wyżej jest użyteczny tylko dla danych dyskretnych. W przypadku danych ciągłych trzeba dokonać ich dyskretyzacji, jest to element kluczowy dla uzyskania optymalnych reguł, np. poprzez analizę histogramów. Analiza histogramowa wykonywana jest dla jednej cechy bez uwzględniania pozostałych, co powoduje, że nie zawsze uzyskuje się optymalny rezultat. Dlatego też w celu zwiększania dokładności oraz umożliwienia automatyzacji procesu wyciągania reguł, wprowadzone zostały do sieci jednostki L. Schemat takiej jednostki przedstawiony jest na poniższym rysunku



Rys 7:3 Jednostka L

Wejście x_i połączone jest poprzez wagi W_1, W_2 do dwóch neuronów, posiadających niezależne progi b_1 oraz b_2. Wagi W_1, W_2 mają wartość ustaloną na +1. Wartość ta nie ulega zmianie podczas uczenia. Funkcje aktywacji wszystkich neuronów są funkcjami sigmoidalnymi lub funkcjami będącymi tangensem hiperbolicznym tanh (4.6), które przy końcu uczenia są bardzo strome. W początkowej fazie natomiast mogą być gładkie, można zacząć np. od skosu 1, co pozwala na rozmytą aproksymacje cech. Dwa neurony jednostki L (*Neuron1*, *Neuron2*) połączone są z trzecim neuronem (*Neuron3*) za pomocą wag *S*1, *S*2. Taka jednostka trenowana wraz z pozostałymi węzłami w sieci powinna wykrywać zmienne lingwistyczne w postaci dwustronnie ograniczonej. Bezpośrednie połączenie jednostek *Neuron1* i *Neuron2* z neuronem wyjściowym jednostki L umożliwia tworzenie przedziałów jednostronnie otwartych.

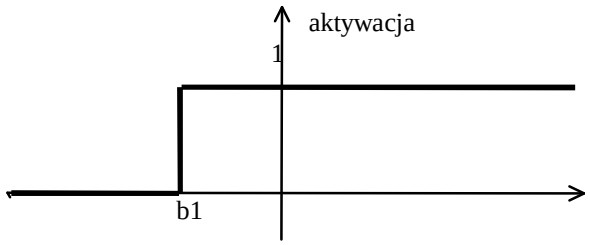
W jednostkach L lepiej jest użyć zamiast funkcji sigmoidalnych funkcji tanh, gdyż tanh przy dużych skosach daje wartości +1, –1 zamiast +1,0. Ma to istotne znaczenie dla liczby reguł otrzymywanych dla poszczególnych węzłów ukrytych. Jednostka L z 3 neuronami ma trzy wyjścia: L_1, L_2, L_3. Funkcja realizowana przez jednostkę L ma postać

$$L(x_i;b_1,b_2,b_3,S_1,S_2)=\begin{cases}\tanh(x_i;b_1) & L_1\\ \tanh(S_1\tanh(x_i;b_1)+S_2(x_i;b_2);b_3) & L_3\\ \tanh(x_i;b_2) & L_2\end{cases} \qquad (5.48)$$

W trakcie uczenia podobnie jak w poprzednim przypadku w funkcji błędu występują dwa dodatkowego człony wymuszające wagi o wartościach +/-1,0. W końcowym etapie nachylenia funkcji aktywacji są bardzo duże, przez co neurony 1,2 i 3 zamieniają się w jednostki progowe, a więc sygnał wychodzący z tych węzłów ma wartość +1 lub –1.

Oczywiście jednostka L może być znacznie bardziej rozbudowana tzn. może wystąpić większa liczba węzłów typu *Neuron3*, a co za tym idzie większa liczba typu *Neuron1*, *Neuron2*. Inicjowanie parametrów adaptacyjnych węzłów w jednostce L następuje tak samo jak w całej sieci tzn. poprzez losowanie liczb z zadanego zakresu.

Neuron1 wzbudza się tylko wtedy, gdy na wejściu pojawia się wartość większa od progu b_1. Węzeł realizuje następującą funkcję.

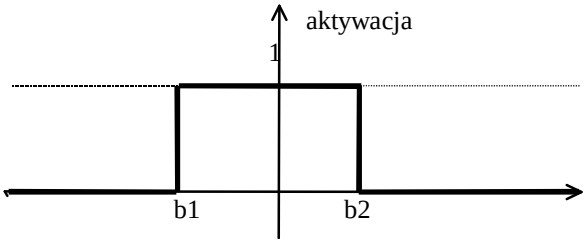


Rys 7:4 Wykres funkcji realizowanej przez *Neuron1*

Podobną funkcję realizuje *Neuron2*, tylko że wartość aktywacji jest równa jeden dla $x_i > b_2$. *Neuron3* natomiast może realizować dowolne złożenie sygnałów z neuronów *Neuron1* i *Neuron2*, przez co powstają różne funkcje w zależności od tego, jakie wartości po zakończeniu uczenia mają wagi S_1, S_2, próg b_3 oraz progi b_1, b_2.

Rozpatrzmy kilka przypadków

1. $S_1=+1$, $S_2=-1$, $2 \geq b_3 > 1$, $b_1 < b_2$

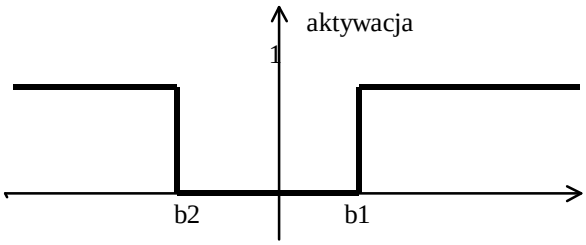


Rys 7:5 Wykres funkcji realizowanej przez *Neuron3*, przypadek 1

otrzymujemy więc zmienną lingwistyczną o wartości 1 w przedziale $x_i \in (b_1, b_2)$ i wartości 0 poza tym przedziałem.

Gdy próg b_3 jest mniejszy od 1 zamiast koniunkcji dwóch węzłów otrzymujemy ich alternatywę, co prowadzi do powstania przedziału $(-\infty, b_2) \cup (b_1, +\infty)$. Tak więc zmienna lingwistyczna $L(x; b_1, b_2, b_3, S_1, S_2)$ uzyskuje wartości $L_3=1$ jeśli x nie należy do przedziału (b_2, b_1)

$b_1 > b_2$



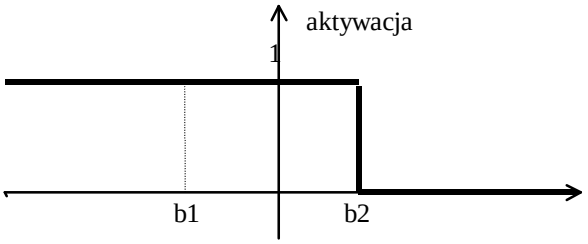
Rys 7:6 Wykres funkcji realizowanej przez *Neuron3*, przypadek 2

Dla odwrotnego przypadku tzn. $b_1<b_2$ otrzymujemy całą przestrzeń, a więc cecha jest niepotrzebna, gdyż węzeł ten będzie wzbudzony dla dowolnego x_i.

Jeśli któraś z wag S_1, S_2 jest zerowa i próg jednostki *Neuron3* $b_3 \leq 1$ otrzymujemy wykres podobny do wykresu przedstawionego na Rys 7:4. W zależności od tego czy druga waga ma wartość dodatnią, czy nie, neuron wzbudza się dla $x_i \in (b, +\infty)$ lub $x_i \in (-\infty, b)$, gdzie b to próg jednostki *Neuron1* lub *Neuron2*.

W pozostałych przypadkach tzn. gdy zachodzi relacja $S_1=S_2=S$, mamy dwie możliwości:

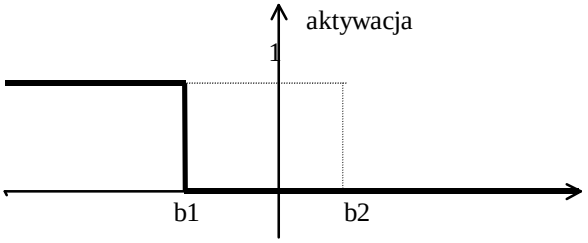
- $S= -1$, $b_3 =1$



Rys 7:7 Wykres funkcji realizowanej przez *Neuron3*, przypadek 3

otrzymujemy więc zmienną lingwistyczną postaci

- $S= -1$, $b_3 =2$



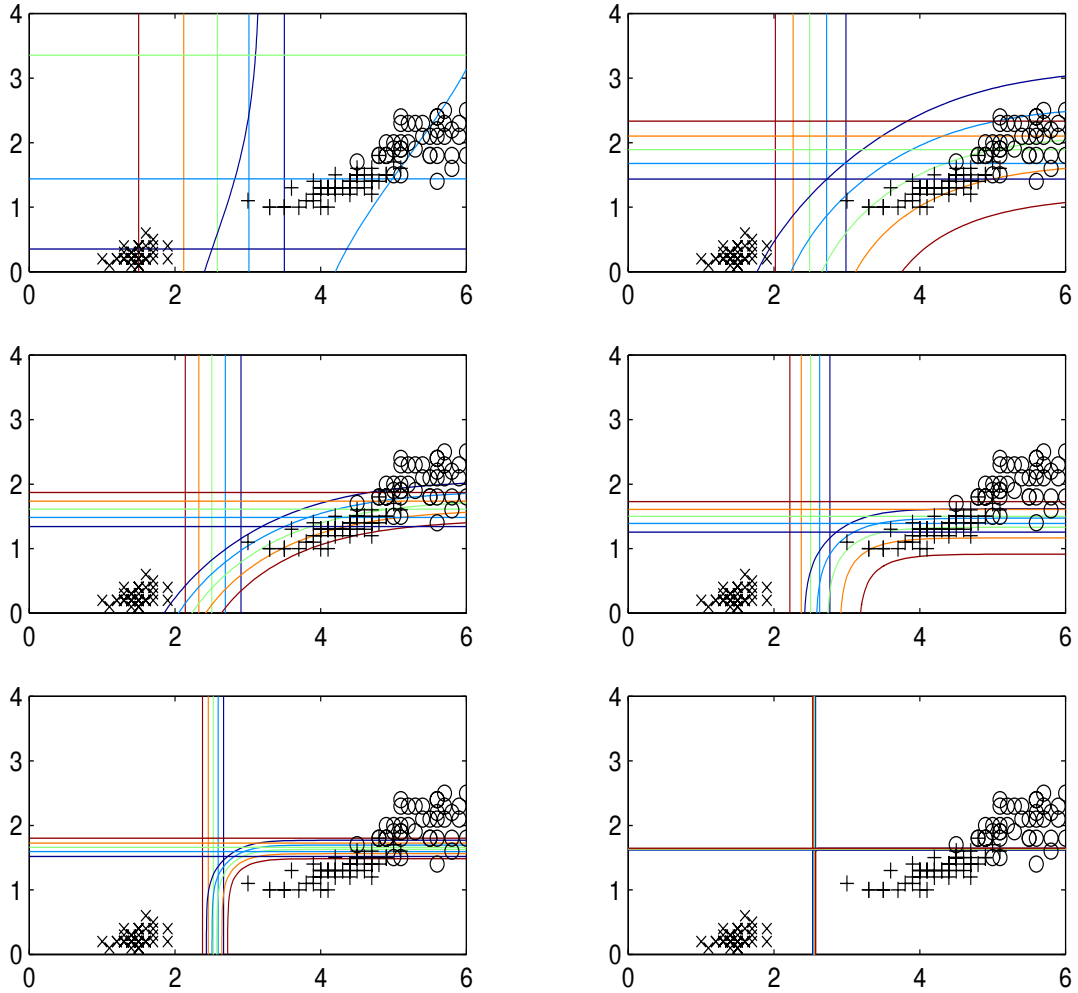
Rys 7:8 Wykres funkcji realizowanej przez *Neuron3*, przypadek 4

Bardzo podobną sytuację otrzymujemy dla warunku $S=1$ z tym, że zamiast przedziałów od $-\infty$ będą przedziały do $+\infty$.

Progi b_1, b_2 definiują granicę zmiennej lingwistycznej, próg b_3 określa czy mamy alternatywę czy koniunkcję sygnałów z węzłów *Neuron1* i *Neuron2*, natomiast wagi są parametrami adaptacyjnymi, które mają na celu ustawienie „zwrotu” sigmoidy.

Końcowa postać sygnału kształtowana jest przez wagi łączące węzły w jednostce L z węzłem ukrytym. W przypadku, gdy waga ma wartość 1, sygnał pozostaje taki jak na wyjściu jednostki L, gdy waga ta ma wartość -1 sygnał zostaje odwrócony.

Przykład stopniowego wyostrzania neuronów w sieci zaprezentowany został na poniższym rysunku.



Rys 7:9 Efekt wyostrzania węzłów w trakcie uczenia

Powyższy rysunek przedstawia zastosowanie sieci MLP2LN do danych *iris*. Dla lepszego zobrazowania procesu wyostrzania wykorzystane zostały tylko cechy 3 i 4, które występują w końcowych regułach. Pierwszy rysunek prezentuje sieć, po inicjalizacji i 20 epokach uczenia, w której funkcje sigmoidalne mają skos 1. Każdy kolejny rysunek przedstawia sieć po nauczeniu : 100, 400, 200, 200 epok. Początkowe obszary zmienności wartości sigmoid są duże, w miarę uczenia są zawężane, a granice decyzyjne przesuwane w taki sposób, aby jak najlepiej odseparować od siebie klasy. Szczególnie trudno jest odseparować klasy *iris virginica* oraz *iris versicolor*. Pierwsza klasa *iris setosa* została odseparowana od pozostałych już na drugim rysunku. Rysunek 4 przedstawia sieć, w której zwiększony został skos do wartości 3, na piątym rysunku wartość skosu wynosiła 5. Ostatni rysunek przedstawia sieć po nauczeniu problemu i ustawieniu skosu o wartości 10000.

7.3. Nieeuklidesowe sieci MLP

Zastosowanie sieci MLP do problemów klasyfikacyjnych prowadzi do dzielenia przestrzenie wejściowej za pomocą „miękkich” hiperpłaszczyzn. Dzięki temu są one podobne do statystycznych metod dyskryminacyjnych, chociaż stosowanie miękkich sigmoid pozwala opisać bardziej złożone obszary. Oczywiście teoretycznie istnieje możliwość otrzymania ostrych

obszarów decyzyjnych, poprzez zwiększenie skosu funkcji sigmoidalnej, jednak prowadzi to do trudności w uczeniu, gdyż nie można stosować metod gradientowych.

Zmiana możliwych kształtów obszarów decyzyjnych realizowanych przez sieć MLP, bez zmiany funkcji transferu oraz algorytmu uczenia, jest możliwa poprzez wykorzystanie związków pomiędzy sieciami neuronowymi a metodami opartymi na podobieństwie. Poprzez zastąpienie ważonej aktywacji kwadratem odległości euklidesowej uzyskuje się sieć MLP opartą na odległości (D-MLP) [15].

Ważona funkcja aktywacji może być przedstawiony w następujący sposób

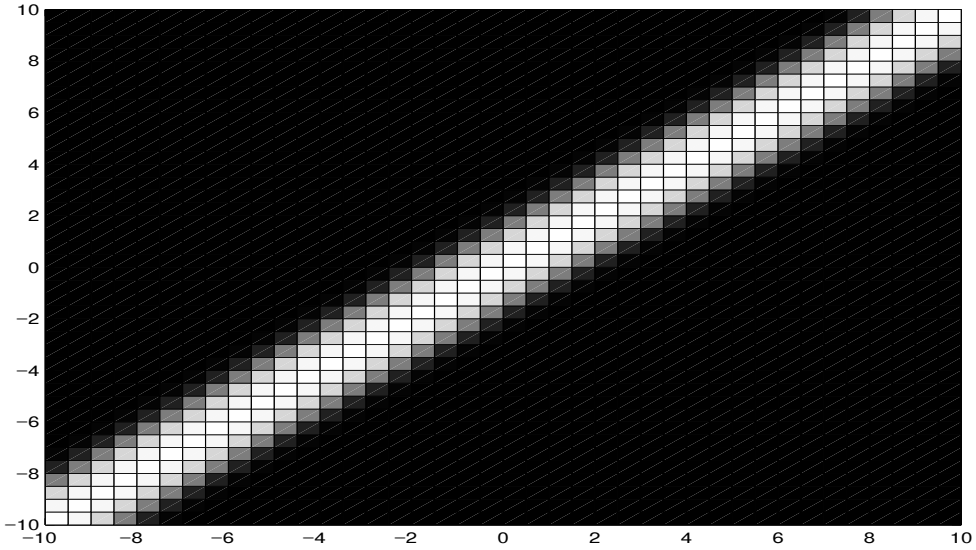
$$\mathbf{wx} = \frac{1}{2}\left(\| \mathbf{w} \|^2 + \| \mathbf{x} \|^2 - \| \mathbf{w} - \mathbf{x} \|^2\right) \tag{5.49}$$

Przy założeniu, że wektory wejściowe **x** są unormowane sigmoidalna funkcje transferu ma postać

$$\sigma(\mathbf{wx} + \theta) = \sigma(p_0 - D(\mathbf{w}, \mathbf{x})) \tag{5.50}$$

gdzie $D(\mathbf{w},\mathbf{x})$ jest kwadratem odległości euklidesowej pomiędzy wektorem wag **w** oraz wektorem wejściowym **x**, czynnik ½ został dołączony do skosu sigmoidy, natomiast $p_0 = \| \mathbf{w} \|^2 + \| \mathbf{x} \|^2$ jest stałym współczynnikiem gdyż normy **x** i **w** są ustalone.

Funkcja transferu $f(D(\mathbf{w},\mathbf{x})) = \sigma(p_0 - D(\mathbf{w},\mathbf{x}))$ staje się funkcją zlokalizowaną, która maleje monotonicznie jako funkcja odległości, z plato dla małych wartości odległości D, osiąga wartość 0.5 dla $D(\mathbf{w},\mathbf{x})=p_0$ i wartość zerową dla dużych odległości.



Rys 7:10 Funkcja transferu $f(p_0\text{-}D(\mathbf{w},\mathbf{x}))$, dla jednowymiarowych wektorów **x**,**w** i odległości euklidesowej $D(\mathbf{w},\mathbf{x})$.

Poprzez zmianę definicji odległości $D(\mathbf{w},\mathbf{x})$ w funkcji transferu f z kwadratu odległości euklidesowej na jakąś inną miarę odległości możliwe jest uzyskanie różnych kształtów decyzyjnych sieci przy niewielkiej jej złożoności. Niewielkiej w porównaniu do sieci, która daje te same kształty decyzyjne, ale przy użyciu funkcji sigmoidalnej.

Norma euklidesowa może zostać zastąpiona na przykład przez normę Minkowskiego. Algorytm wstecznej propagacji wymaga istnienia pochodnej funkcji odległości, co dla normy Minkowskiego jest spełnione.

Uogólniona odległość Minkowskiego z czynnikiem skalującym dana jest przez następujący wzór:

$$D(\mathbf{A},\mathbf{B};\mathbf{s})^{\beta} = \sum_{j}^{N} s_j d(A_j, B_j)^{\alpha} \tag{5.51}$$

gdzie zazwyczaj $\alpha=\beta$. Funkcja $d(\cdot)$ ocenia podobieństwo na poziomie cech i w najprostszym przypadku ma postać $d(A_j, B_j) = | A_j - B_j |$. Dla $\alpha=\beta=2$ wektory $\| \mathbf{A} \|=1$ leżą na sferze o promieniu 1, dla dużych α sfera zmienia się w miękki sześcian, dla $\alpha=1$ ma kształt piramidy.

Zmiana definicji odległości na nieeuklidesową zmienia całkowicie kształt obszarów decyzyjnych z hiperpłaszczyzn w obszary sferyczne czy też kubiczne. Wyznaczenie pochodnej funkcji transferu dla normy Minkowskiego jest proste, wymaga jednak modyfikacji w standardowym algorytmie propagacji wstecznej.

Możliwe jest zastosowanie sieci nieeuklidesowych bez jakiejkolwiek zmiany w standardowym algorytmie propagacji wstecznej, a jedynie poprzez odpowiednią transformację danych treningowych.

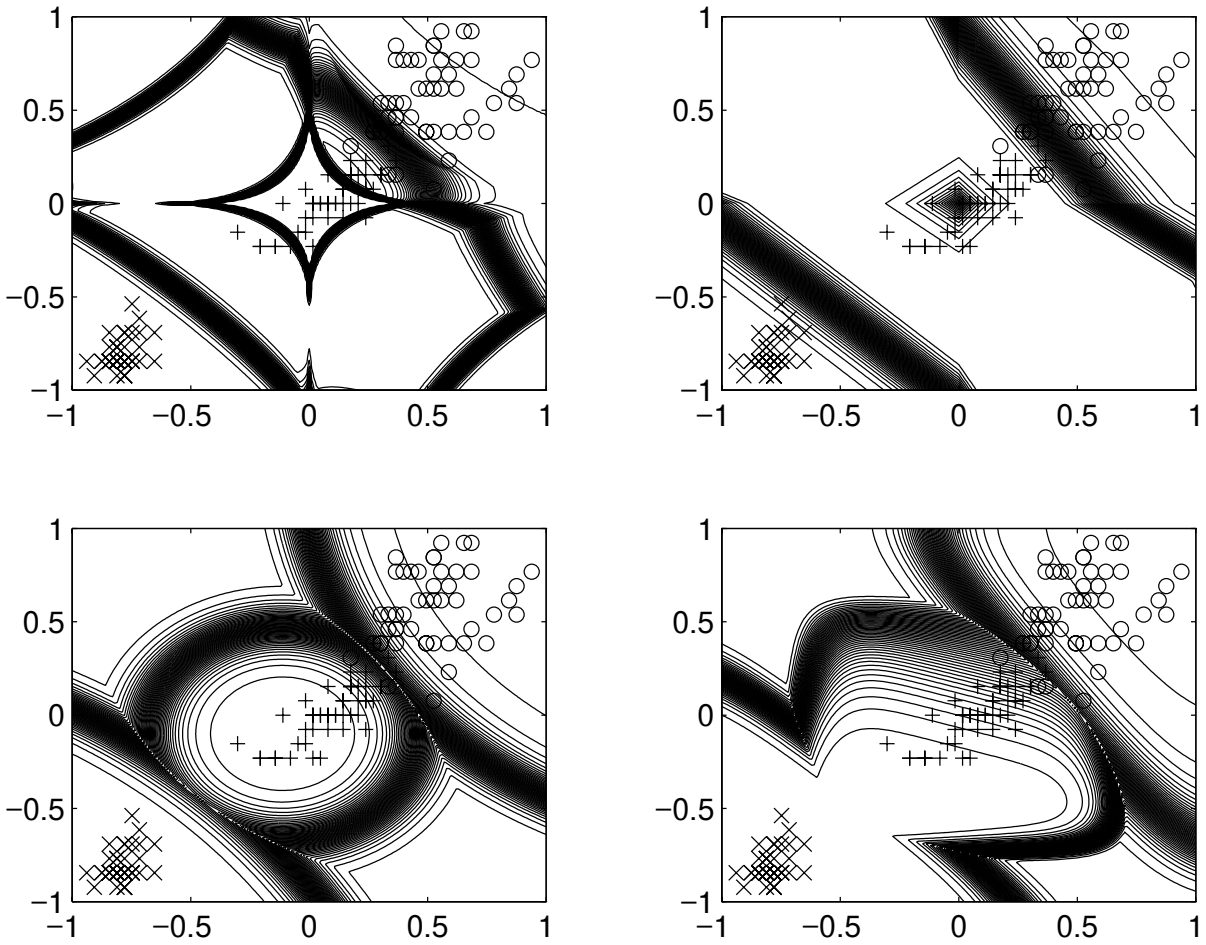
W sieciach nieeuklidesowych wymagane jest, aby norma wektorów wejściowych była ustalona, czyli $\| \mathbf{x} \|= q$ dla wszystkich **x**. Utworzenie danych z ustaloną normą bez straty informacji jest możliwe poprzez dodanie dodatkowego składnika do wektora treningowego, powiększając w ten sposób przestrzeń cech, o co najmniej jeden wymiar. Wzięcie $x_r = \sqrt{R^2 - \| \mathbf{x} \|^2}$, gdzie $R \geq \max_{\mathbf{x}} \| \mathbf{x} \|$ powoduje umieszczenie danych na półsferze o promieniu *R*. W ogólności wektory postaci $(\mathbf{x}, x_r)$ mogą zostać znormalizowane poprzez odpowiednio zdefiniowaną funkcje odległości. Niestety w przypadku, gdy w danych występuje niewielka liczba wektorów o dużej normie wykonanie rzutowania powoduje bardzo nierównomierny ich rozkład na sferze. Większość danych ma małą współrzędną x_r w związku z czym leżą blisko dolnej części półsfery, co znacznie utrudnia proces uczenia sieci. Aby uniezależnić się od skal w poszczególnych cechach (co może powodować, że jedna cecha ma decydujące znaczenie dla wartości normy) wykonywana jest standaryzacja, a następnie przeskalowanie wartości cech tak, aby maksymalna wartość była +1, a minimalna –1. Mimo to niekiedy mogą wystąpić problemy dużej normy szczególnie dla danych zawierających dużą liczbę wymiarów. W takich przypadkach trzeba najpierw wykonać selekcję cech.

Poniższy rysunek przedstawia różne kształty granic decyzyjnych dla danych iris. Dane zawierają trzy klasy: setosa, versicolor, virginica. Każda z tych klas opisana jest przez 50 przypadków. Wszystkie wektory składają się z 4 współrzędnych: długość i szerokość liścia oraz długość i szerokość płatka. W tym przypadku zostały użyte tylko dwie współrzędne. W dwu wymiarowej przestrzeni klasy virginica i versicolor nakładają się na siebie, dlatego też idealne odseparowanie tych klas nie jest możliwe.

Standardowa sieć MLP do rozwiązania tego problemu potrzebuje 4 jednostek ukrytych i 3 jednostek wyjściowych reprezentujących poszczególne klasy. Zastosowanie sieci nieeuklidesowej umożliwia znalezienie rozwiązania przy 3 jednostkach ukrytych i 3 wyjściowych.

Dane zostały zestandaryzowane i przeskalowane tak, aby maksymalna wartość dla danej cechy wynosiła 1, a minimalna –1. Następnie dodana została dodatkowa współrzędna w taki sposób, aby każdy z wektorów był znormalizowany, w sensie odpowiedniej miary odległo-

ści Minkowskiego. Poprzez zastosowanie różnych wartości wykładnika α miary odległości Minkowskiego, otrzymuje się różne obszary decyzyjne np. w kształcie okręgu (dla miary euklidesowej, czyli α=2), kształt romboidalny α=1, prostokątne obszary decyzyjne (pozwalające tworzyć reguły logiczne) dla α o dużej wartości, lub też cykloidalne dla α<1. Dla danych iris optymalne rozwiązanie (3 błędy) uzyskuje się dla α>0.8.



Rys 7:11 Obszary decyzyjne dla nieeuklidesowych sieci MLP, przy użytej normie Minkowskiego, α=0.5,1,2,7

8. FSM

Model FSM (Feature Space Mapping) [18] oparty jest na estymacji gęstości rozkładu prawdopodobieństwa prezentowanych danych. Podobnie jak sieć RAN oraz RecBFN, FSM jest siecią z algorytmem konstruktywistycznym.

Patrząc z różnych perspektyw model FSM można uważać za sieć neuronową podobną do sieci z radialnymi funkcjami bazowymi (chociaż funkcje nie musza być radialne, tylko separowalne), system neurorozmyty (aktywność węzłów sieci z separowalnymi funkcjami transferu można przeanalizować używając koncepcji zbiorów rozmytych), rodzaj systemu opartego na śladach pamięci (memory-based system), system samoorganizujący się a nawet jądro systemu eksperckiego, wykorzystującego wiedzę rozmytą lub oferującego przydatne heurystyki w procesie szukania rozwiązań. Składowe wektora wejściowego i wyjściowego definiują razem przestrzeń cech. Na podstawie danych treningowych, stanowiących prototypy w przestrzeni cech, tworzy się rozmyte obiekty reprezentujące łączny rozkład prawdopodobieństwa dla wektorów wejściowych i wyjściowych. Rozkład ten może być użyty do aproksymacji nieznanych zależności lub do klasyfikacji.

Zastosowanie funkcji separowalnych pozwala na przeszukiwanie przestrzeni cech pomimo występowania wartości brakujących. Umożliwia to wyszukanie węzła najbardziej odpowiadającego aktualnym danym wejściowym, jak również dopełnienie brakującej wartości poprzez uzupełnienie faktu na podstawie teorii opisywanej przez wyszukany węzeł. Wiedza zapisana w przestrzeni cech jest używana w procesie przeszukiwania w taki sam sposób jak wiedza heurystyczna używana jest przez eksperta do zmniejszenia przestrzeni poszukiwań.

Model FSM może być również zastosowany do odkrywania wiedzy, praw, które mogą zostać wydedukowane z przykładów. Prawa te mogą być reprezentowane przez funkcje gaussowskie, wówczas rozmycie funkcji gaussowskiej odpowiadającej danemu faktowi jest miarą zaufania do tego faktu, gdzie zaufanie bazuje na liczbie zaprezentowanych przykładów.

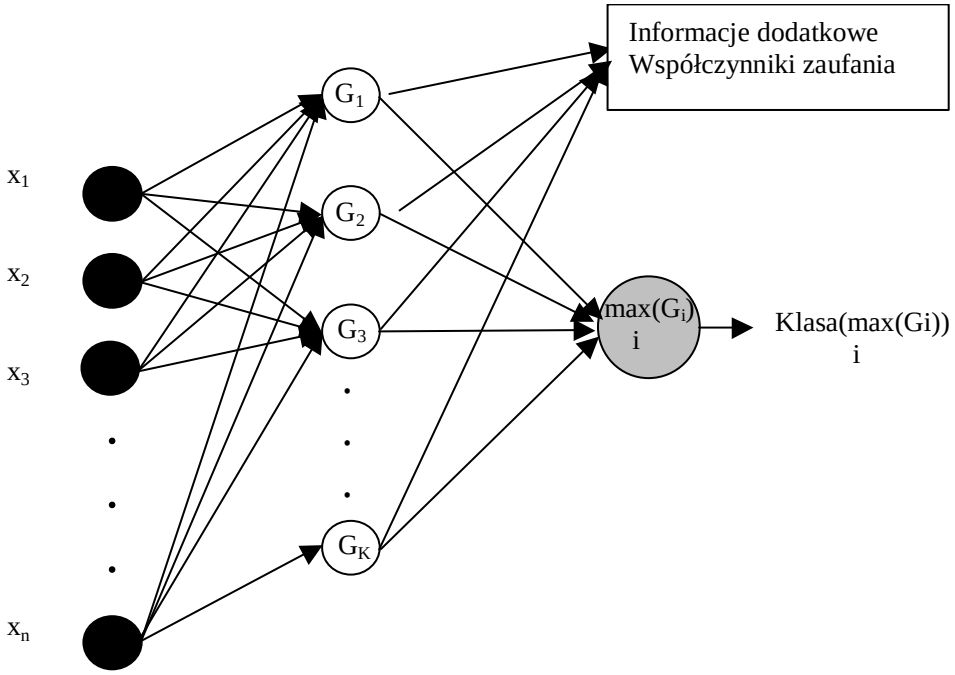
Topograficzna struktura przestrzeni cech może podlegać ograniczeniom związanym z symboliczną wiedzą aprioryczną (zależności pomiędzy elementami wektora wejściowego). Szkielet tej struktury tworzy się korzystając z metod inicjalizacji opartych na klasteryzacji, a szczegóły ustalają się w wyniku procesu uczenia. W pewnym sensie jest to więc meta-model, gdyż możliwa jest różna specyfikacja jego matematycznych aspektów. Pierwotną inspiracją do jego stworzenia była próba wprowadzenia pośredniego poziomu opisu funkcji poznawczych jako przybliżenia do neurodynamiki: wysoka aktywność neuronów wysyłających impulsy opisywana jest przez dużą gęstość prawdopodobieństwa w tym obszarze przestrzeni, który odpowiada kombinacji cech koniecznych do wywołania takiej aktywności.

Z teoretycznego punktu widzenie model FSM nadaje się zarówno do problemów aproksymacyjnych jak i czysto klasyfikacyjnych. W tej pracy przedstawiony jest algorytm, który dotyczy tylko zagadnienia klasyfikacji.

8.1. Architektura sieci FSM

W sieci FSM występują tylko trzy warstwy neuronów: wejściowa, ukryta i wyjściowa. Liczba węzłów w warstwie wejściowej jest ustalona na początku procesu uczenia i jest równa wymiarowi wektorów wejściowych. W warstwie wyjściowej wystarczy tylko jeden węzeł (można też używać jednego węzła na jedną klasę), którego zadaniem jest określenie na podstawie wzbudzeń neuronów w warstwie ukrytej, do której klasy należy wektor wejściowy **x**.

Węzły warstwy ukrytej połączone są ze wszystkimi neuronami warstwy wejściowej oraz z węzłem wyjściowym. Sieć FSM może mieć strukturę modułową, składającą się z kilku podsieci specjalizujących się w klasyfikacji danych wejściowych określonych przez cechy jednego typu.



Rys 8:1 Architektura sieci FSM

Wszystkie węzły w warstwie ukrytej realizują dowolną funkcję faktoryzowalną $G(\mathbf{x};\mathbf{D},\sigma)$.

$$G(\mathbf{x};\mathbf{D},\sigma)=\prod_i G_i(x_i;D_i,\sigma_i)$$

gdzie każdy ze składników sparametryzowany jest przez położenie i dodatkowy parametr, który dla funkcji zlokalizowanych pełni rolę dyspersji.

Chociaż teoretycznie nie ma innych ograniczeń na funkcje aktywacji w FSM, w praktyce jednak na razie stosowane były tylko funkcje zlokalizowane, o wartościach przeskalowanych do przedziału [0,1]. Zmiana wartości aktywacji w miarę oddalania się od centrum może być również skokowa, co ma miejsce w przypadku użycia funkcji prostokątnej.

Funkcja realizowana przez sieć FSM ma postać

$$FSM(\mathbf{x})=Klasa\left[\max_i\left(G_i(\mathbf{x};\mathbf{D},\sigma)\right)\right] \tag{5.52}$$

indeks *i* ma wartość od 1 do liczby węzłów w sieci.

Celem uczenie sieci FSM jest minimalizacja funkcji błędu postaci (1.5). Proces uczenia trwa tak długo, aż sieć osiągnie żądaną jakość klasyfikacji ρ.

8.2. Dane brakujące

Zastosowanie separowalnych funkcji transferu umożliwia omijanie wartości brakujących podczas uczenia jak również ich dopełnianie. Omijanie wartości brakujących polega na pomijaniu podczas obliczania aktywacji funkcji transferu wymiaru, który zawiera wartość brakującą, lub inaczej w tym wymiarze wzbudzenia osiągnie wartość maksymalną, czyli 1. Dla danego wektora wejściowego zawierającego wartość brakującą wszystkie istniejące węzły będą miały w tym wymiarze to samo wzbudzenie. W przypadku, gdy wektor ma wszystkie wartości brakujące wszystkie węzły mają wartość aktywacji równą 1, czyli wektor należy do wszystkich istniejących klas, gdyż nie można nic o nim powiedzieć. Metoda ta nadaje się jednak tylko do danych, które nie zawierają dużej liczby wartości brakujących. W przeciwnym wypadku mogą powstawać węzły, które mają niewielką liczbę aktywnych wymiarów, przez co bardzo często ulegają wzbudzaniu.

Dopełnianie wartości brakujących polega na uzupełnieniu braków na podstawie istniejącej sieci. W przypadku danych testowych jest to naturalne, gdyż uzupełnienie następuje poprzez dobranie wartości brakującej w taki sposób, aby zmaksymalizować wzbudzenie węzła zwycięskiego, czyli po prostu wzięcie odpowiednich współrzędnych położenia tego węzła lub też, aby zmaksymalizować współczynnik zaufania postaci

$$p(\mathbf{x}) = \frac{\sum_k G(\mathbf{x}; D_k^{C_i}, \sigma_k^{C_i})}{\sum_j G(\mathbf{x}; D_j, \sigma_j)} \tag{5.53}$$

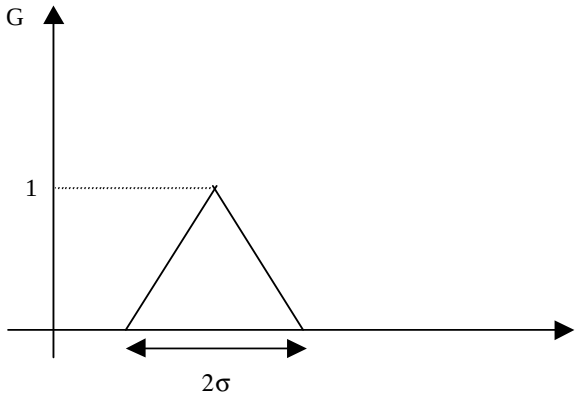
C_i jest klasą reprezentowaną przez neuron zwycięski, k indeks przebiegający po węzłach z klasy C_i, j indeks przebiegający po wszystkich węzłach w warstwie ukrytej sieci.

W przypadku danych treningowych na początku odrzuca się wszystkie wektory zawierające braki, oczywiście przy założeniu, że nie ma ich dużo, a następnie uczy się sieć na pozostałych wektorach. Po nauczeniu następuje uzupełnienie wartości brakujących odrzuconych wektorów a następnie ponowne nauczenie sieci na pełnych danych.

8.3. Funkcje transferu

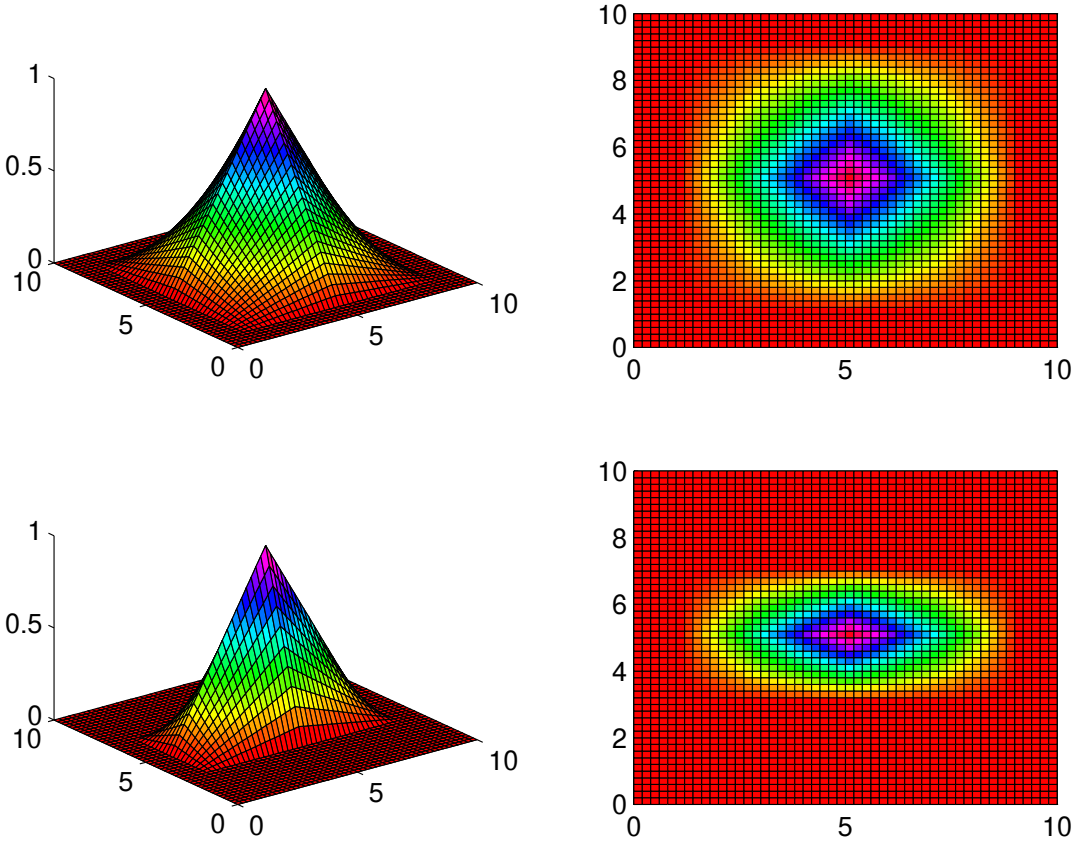
Dotychczas w FSM stosowane były następujące funkcje transferu:

- Trójkątna



Rys 8:2 Jednowymiarowa funkcja trójkątna

$$G(x_i; D_i, \sigma_i) = \begin{cases} \dfrac{-|x_i - D_i| + \sigma_i}{\sigma_i} \; jeżeli |x_i - D_i| \le \sigma_i \\ 0 \;\; w\ przeciwnym\ przypadku \end{cases} \tag{5.54}$$



Rys 8:3 Obszary decyzyjne dla dwu wymiarowej funkcji trójkątnej. Na rysunku pierwszym rozmycia w poszczególnych wymiarach są jednakowe, na drugim różne.

- Bicentralna [20]

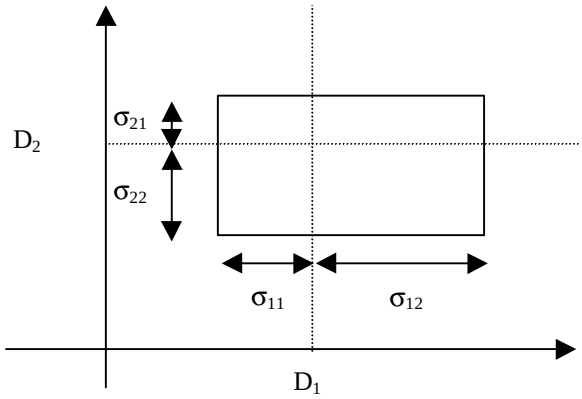
$$G(\mathbf{x}; \mathbf{D}, \sigma_1, s_1, \sigma_2, s_2) = \prod_i sig(x_i - D_i, \sigma_{1i}, s_1)(1 - sig(x_i - D_i, \sigma_{1i}, s_1)) \tag{5.55}$$

Ponieważ funkcja bicentralna może mieć maksymalną wartość poniżej 1, to w zastosowaniu do sieci FSM musi zostać przenormalizowana.

- Prostokątna

$$G(x_i; D_i, \sigma_i) = \begin{cases} 1 : | x_i - D_i | \leq \sigma_i \\ 0 : w\ przeciwny\ razie \end{cases} \tag{5.56}$$

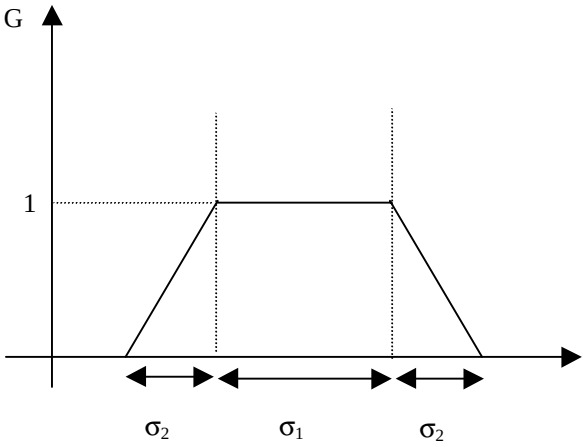
- Prostokątna niesymetryczna



Rys 8:4 Funkcja prostokątna niesymetryczna

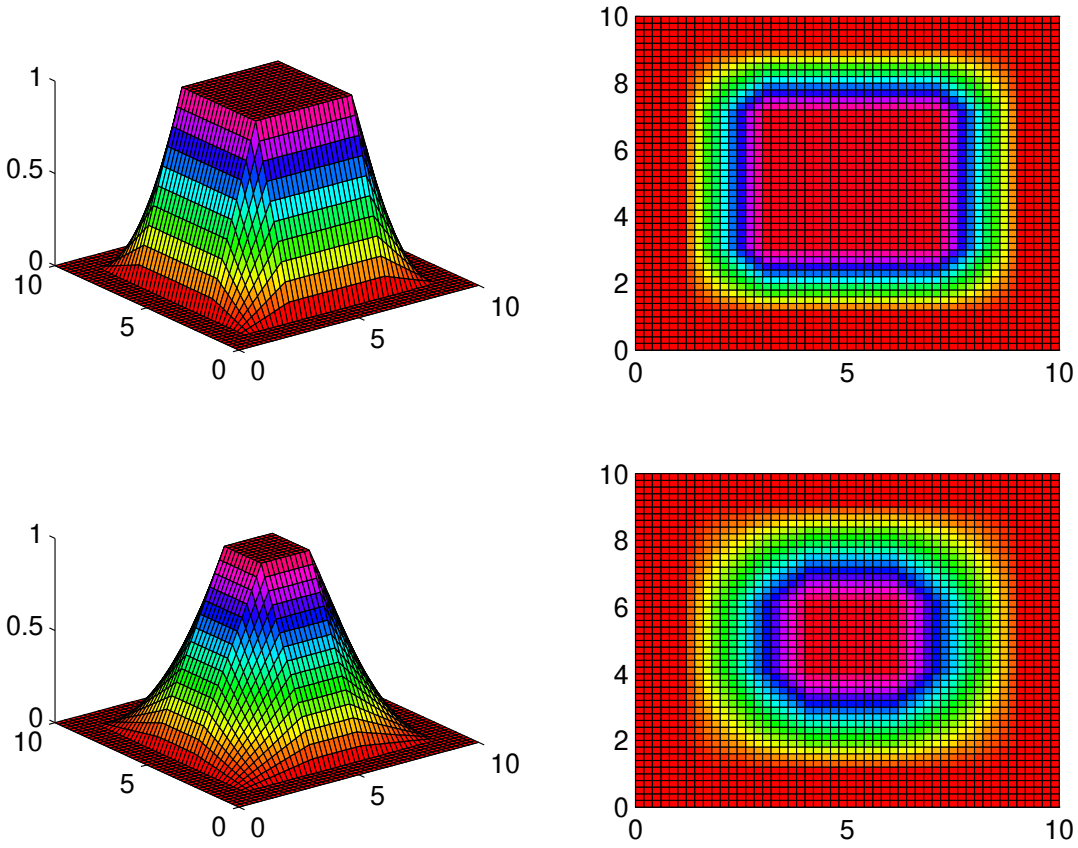
$$G(x_i; D_i, \sigma_1, \sigma_2) = \begin{cases} 1 : x_i - D_i < 0 \wedge D_i - x_i < \sigma_{i1} \\ 1 : x_i - D_i >= 0 \wedge x_i - D_i < \sigma_{i2} \\ 0 : w\ przeciwnym\ wypadku \end{cases} \tag{5.57}$$

- Trapezoidalna



Rys 8:5 Schemat funkcji trapezoidalnej w przestrzeni jednowymiarowej

$$G(x_i; D_i, \sigma_i) = \begin{cases} 1 & gdy\,|\, x_i - D_i \,|< \sigma_{i1} \\ \dfrac{|\,(|\, x_i - D_i \,| - \sigma_{i1})\,| + \sigma_{i2}}{\sigma_{i2}} & gdy\,\sigma_{i2} + \sigma_{i1} >|\, x_i - D_i \,|> \sigma_{i1} \\ 0 & w\ przeciwnym\ przypadku \end{cases} \tag{5.58}$$



Rys 8:6 Obszary decyzyjne dla dwu wymiarowej funkcji trapezoidalnej. Na rysunku pierwszym rozmycia w poszczególnych wymiarach są jednakowe, na drugim różne.

- Gaussowska postaci (5.7)

Podczas obliczania aktywacji węzłów, przy zastosowaniu dowolnej funkcji aktywacji, istnieje możliwość określenia progu minimalnej aktywacji υ. Wówczas

$$G'(\mathbf{x}; \mathbf{D}, \sigma) = \begin{cases} G(\mathbf{x}; \mathbf{D}, \sigma) : G(\mathbf{x}; \mathbf{D}, \sigma) \geq \upsilon \\ 0 : w\ przeciwnym\ wypadku \end{cases} \tag{5.59}$$

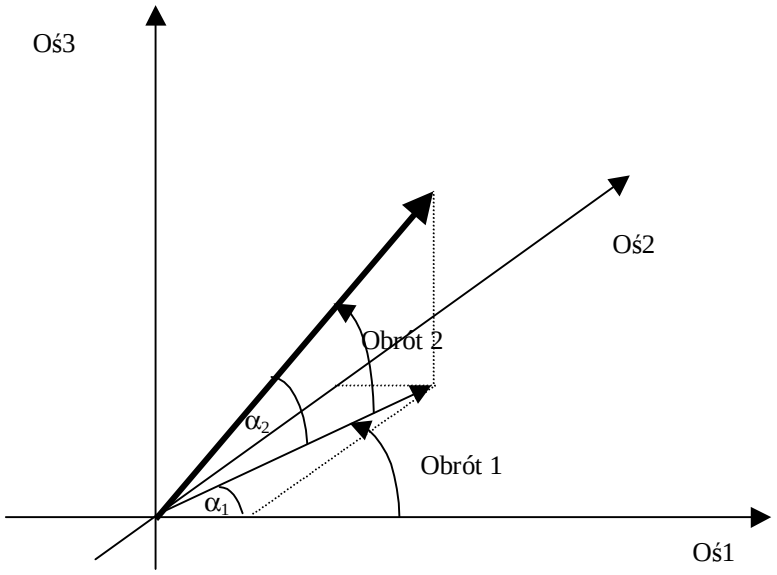
Podczas uczenia próg minimalnej aktywacji υ jest zmniejszany zgodnie z następującą relacją

$$\upsilon = \begin{cases} \upsilon : \dfrac{\tau}{\tau_\upsilon}\ nie\ jest\ calkowite \\ \dfrac{\upsilon}{2} : w\ przeciwnym\ razie \end{cases} \tag{5.60}$$

W celu lepszego dopasowania węzłów sieci do danych w FSM stosuje się neurony, których funkcje transferu w *n* wymiarowej przestrzeni mają możliwość obrotu realizowanych przez nie obszarów decyzyjnych. Wykonanie pełnego obrotu jest możliwe np. poprzez użycie funkcji gaussowskiej postaci (5.8) jest to jednak obliczeniowo bardzo kosztowne i powoduje gwałtowne zmniejszenie się szybkości uczenia. Dlatego też zastosowano obroty będące złożeniem obrotów dwuwymiarowych.

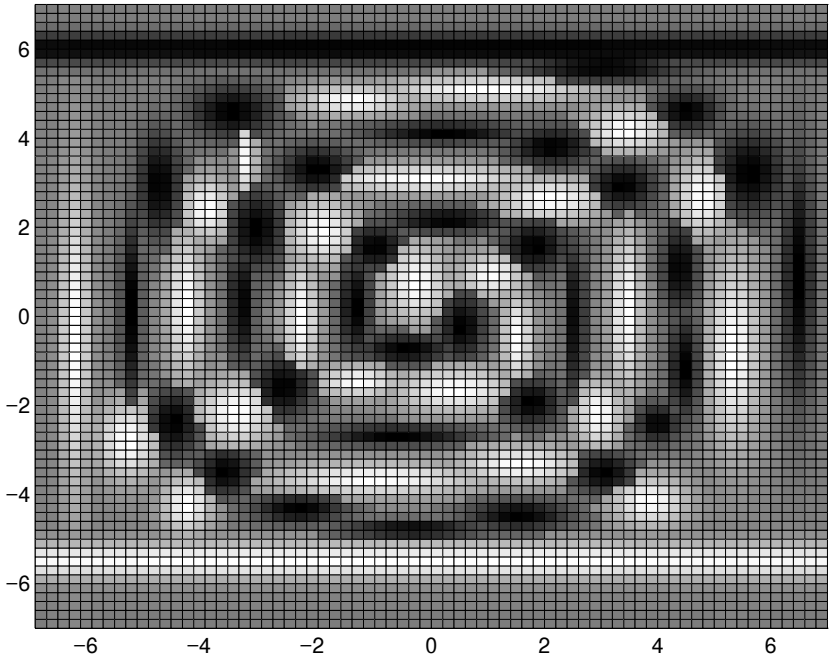
$$\begin{pmatrix} x_1' \\ . \\ . \\ . \\ x_n' \end{pmatrix} = \begin{pmatrix} \cos(\alpha) & 0 & \cdots & -\sin(\alpha) & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ 0 & \cdots & \ddots & \vdots & \vdots \\ \sin(\alpha) & 0 & \cdots & \cos(\alpha) & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ . \\ . \\ . \\ x_n \end{pmatrix} \tag{5.61}$$

Obrót wykonywany jest w taki sposób, aby oś układu opisująca pierwszą składową znalazła się w pozycji wskazującej przez poszczególne kąty obrotu $\alpha_1, \alpha_2, \ldots, \alpha_{n-1}$, gdzie α_1 jest kątem pomiędzy zadanym wejściowym wektorem x_1 (zaczepionym w początku układu współrzędnych danego węzła), a osią opisującą pierwszą składową w tym układzie, α_2 kąt pomiędzy zadanym wektorem a płaszczyzną 1-2 itd. Trójwymiarowy przykład przedstawiony jest na poniższym rysunku.

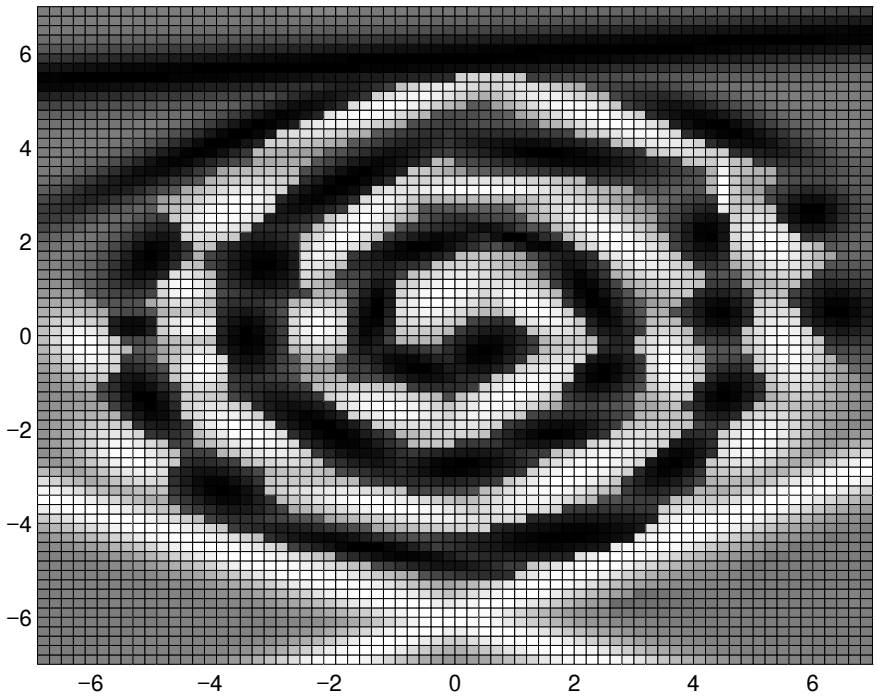


Rys 8:7 Trójwymiarowy przykład złożenia dwu wymiarowych obrotów

Poniższe rysunki przedstawiają działanie sieci FSM z funkcjami gaussowskimi bez obrotów oraz z obrotami dla problemu dwóch spiral. W problemie tym występuje 196 dwu wymiarowych wektorów, które reprezentują dwie klasy po 98 punktów każda. Klasy zawierają punkty przedstawiające spirale wzajemnie przenikające się. Celem jest takie nauczenie sieci, aby w jak najlepiej odseparować klasy od siebie.



Rys 8:8 Rezultat działania sieci FSM z funkcjami gaussowskimi bez obrotu



Rys 8:9 Rezultat działania sieci FSM z funkcjami gaussowskimi z obrotem

W obu przypadkach liczba powstałych węzłów jest podobna (52). Obszar decyzyjny uzyskany przez sieć FSM, w której wykorzystane zostały obroty, wydaje się lepiej opisywać dane spirali. Na rysunku 8:9 widać wyraźnie obszary powstałe z obrócenia funkcji aktywacji.

Najlepsze rezultaty zarówno pod względem liczby powstałych węzłów jak i generalizacji uzyskiwane były dla funkcji gaussowskiej. Wybór funkcji aktywacji zależny jest od celu, który chcemy uzyskać, np. do stworzenia klasyfikatora, którego działanie możemy w prosty sposób prześledzić, najlepiej nadają się reguły logiczne, które w FSM można uzyskać stosując funkcje prostokątne. Używając natomiast funkcji gaussowskiej czy też trójkątnej można rów-

nież uzyskać reguły, są to jednak reguły rozmyte, których interpretacja jest znacznie trudniejsza od klasycznych reguł logicznych. Każdy węzeł ukryty charakteryzowany jest przez położenie **D**, i rozmycie σ, które potrzebne są do wyliczenia funkcji aktywacji. W przypadku niektórych funkcji aktywacji np. niesymetrycznej funkcji prostokątnej występuje większa liczba parametrów adaptacyjnych (dwa rozmycia dla każdego z wymiarów), co powoduje niewielkie modyfikacje w algorytmie uczenia.

Występują jeszcze wielkości, które nie mają wpływu na wartość wzbudzenia, ale są potrzebne w trakcie uczenia do określenia wielkości zmian pozostałych parametrów. Są to:

- masa *m* określająca liczbę wektorów klasyfikowanych przez dany węzeł
- czas powstania τ_n, czyli numer epoki, w której dany węzeł powstał
- klasa, którą węzeł reprezentuje

8.4. Inicjalizacja sieci FSM

Ponieważ algorytm uczenia w FSM jest algorytmem konstruktywistycznym, to można przyspieszyć proces uczenia poprzez wstępne oszacowanie liczby węzłów ukrytych i odpowiednie ich zainicjowanie. Oszacowanie to powinno być raczej optymistyczne tzn. liczba węzłów utworzona podczas inicjalizacji powinna być zaniżona. W przeciwnym wypadku wstępna inicjalizacja prowadzi do zbytniego rozdrobnienia i nawet po użyciu optymalizacji nie uzyska się dobrych rezultatów.

Dzięki stosowaniu funkcji zlokalizowanych do ustalenia liczby węzłów ukrytych jak i ich zainicjowania można użyć metod opartych na grupowaniu. W wyniku klasteryzacji otrzymujemy zbiór skupisk, prototypów, w środku których ustawiane są węzły sieci.

$$D_i^k = \frac{1}{n^k} \sum_{j=1}^{n^k} x_{ji}^k \tag{5.62}$$

gdzie n^k liczba wektorów znajdujących się w *k*-tym klastrze.

Rozmycia tych prototypów ustalane są na podstawie wielkości odpowiadających im klastrów.

$$\sigma_i^k = \frac{\max(x_i^k) - \min(x_i^k)}{2} \tag{5.63}$$

Gdy któreś z rozmyć ma wartość zero przyjmowana jest, jeżeli jest niezerowa, wartość $def_{\sigma i}$ postaci

$$def_{\sigma i} = \frac{1}{l} \sum_{p=1}^{l} \sigma_i^p \tag{5.64}$$

gdzie indeks *p* przebiega po wszystkich skupiskach reprezentujących tą samą klasę co skupisko *k*, *l* liczba skupisk z tej samej klasy co *k*. W przypadku gdy $def_{\sigma i}$ ma wartość zero przyjmowana jest minimalna wartość nie zerowa.

Z uwagi na to, że funkcja aktywacji jest iloczynem po poszczególnych wymiarach, jeśli choć jedna ze składowych ma wartość 0, to cała funkcja aktywacji jest zerowa. Tak ustalone rozmycia zostają następnie proporcjonalnie zwiększone, po to, aby sieć w początkowej fazie uczenia od razu klasyfikowała, bez względu na poprawność, wszystkie wektory z ciągu treningowego. Dzięki temu unika się przypadkowego wstawiania węzłów na początku uczenia.

W przypadku użycia funkcji transferu z obrotem, będącym złożeniem obrotów dwuwymiarowych, podczas inicjalizacji kąty nachylenia klastrów znajdowane są za pomocą prostej $x_i = a_i x_1$ dopasowanej na podstawie wszystkich wektorów należących do tego skupiska. Stąd poszczególne parametry

$$a_i = \frac{\sum_l x_1^l x_i^l}{\sum_l x_i^l x_i^l} \tag{5.65}$$

Współczynniki te równe są tangensowi kąta pomiędzy główną osią skupiska a osią Oś1. Tak więc poszczególne kąty można wyznaczyć ze wzoru

$$\phi_k = \arctan(a_k) \tag{5.66}$$

8.5. Algorytm uczenia

Algorytm uczenia sieci FSM podobnie jak algorytm DDA, czy też RAN polega na budowaniu sieci od zera tzn. na początku występują tylko węzły wejściowe i wyjściowe. Na wejście sieci wprowadzane są w sposób losowy wektory ze zbioru treningowego. Na podstawie podobieństwa między wektorem wejściowym a istniejącymi węzłami w sieci, następuje podjęcie decyzji o dostawieniu nowego węzła lub też dostrajaniu parametrów adaptacyjnych węzłów w sieci.

Po zakończeniu każdej epoki następuje sprawdzenie klasyfikacji sieci. Jeżeli jakość klasyfikacji jest mniejsza od żądanej ρ, następuje sprawdzenie poprawności klasyfikacji Q wszystkich węzłów znajdujących się w warstwie ukrytej. Jeżeli dla danego węzła $Q < \rho$, to następuje zmniejszenie rozmycia

$$\sigma = \sigma - \lambda(1-Q)\sigma \tag{5.67}$$

gdzie

$$\lambda = \begin{cases} 0.3 : \tau_n < 5 \\ 0.9 : \tau_n \geq 5 \end{cases} \tag{5.68}$$

Jeżeli po zakończeniu epoki jakość klasyfikacji sieci jest większa lub równa żądanej, to uczenie jest przerywane i następuje usuwanie węzłów zbędnych tzn. takich, których usunięcie nie powoduje zmniejszenia jakości klasyfikacji sieci lub też takich, które klasyfikują niewielką liczbę wektorów. Następuje ponowne uczenie sieci. Proces uczenie i douczania powtarzany

jest kilkakrotnie. Po zakończeniu douczania usuwane są ostatecznie małe węzły i etap uczenia zostaje zakończony.

8.5.1. Podjęcie decyzji o dostawieniu lub optymalizacji neuronu

Jeśli wektor wejściowy jest dostatecznie podobny do istniejącego węzła wystarczy zmodyfikować parametry opisujące ten węzeł. Podobieństwo wyznaczane jest na podstawie odległości euklidesowej między wektorem wejściowym a poszczególnymi neuronami $S_1(\mathbf{x}^i)$ w sieci, lub też na podstawie pobudzenia węzłów $S_2(\mathbf{x}^i)$.

$$S_1(\mathbf{x}^i) = \begin{cases} 1 : Klasa(\min_k \| \mathbf{x}^i - \mathbf{D}_k \|) = C^i \\ 0 : w\ przeciwnym\ wypadku \end{cases} \tag{5.69}$$

gdzie $\|\cdot\|$ oznacza normę euklidesową, *i* indeks wektora wejściowego, *k* indeks węzła w warstwie ukrytej

$$S_2(\mathbf{x}^i) = \begin{cases} 1 : Klasa(\max_k (G(\mathbf{x}^i; \mathbf{D}_k, \sigma_k))) = C^i \\ 0 : w\ przeciwnym\ wypadku \end{cases} \tag{5.70}$$

Jeżeli $S(\mathbf{x}^i) = 1$, to dokonywana jest optymalizacja parametrów węzła, który jest najbliższy w sensie euklidesowym do aktualnego wektora wejściowego. Jeżeli $S_1(\mathbf{x}^i) = 0$, to sprawdzany jest $S_2(\mathbf{x}^i)$. Jeżeli $S_2(\mathbf{x}^i) = 1$, to przeprowadzana jest optymalizacja węzła najbardziej wzbudzającego się, w przeciwnym razie następuje dostawienie nowego neuronu.

8.5.2. Nowy neuron

Po dostawieniu nowego neuronu parametry tego neuronu ustawiane są w następujący sposób

$$D_i = x_i \tag{5.71}$$

$$\tau_n = \tau \tag{5.72}$$

$$m = 1 \tag{5.73}$$

$$\sigma_i = \begin{cases} \min_k(|\ D_i - D_i^k\ |) : \min_k(|\ D_i - D_i^k\ |) > 0 \\ \sigma_i = \sigma_i^l : w\ przeciwnym\ razie \end{cases} \tag{5.74}$$

indeks *l* oznacza wybrany neuron z warunku $\min_k |\ D_i - D_i^k\ |$.

8.5.3. Optymalizacja parametrÛw adaptacyjnych neuronÛw

W przypadku, gdy dla danego wektora wejściowego $\mathbf{x}^i$ nowy neuron nie jest dostawiony, następuje optymalizacja parametrów adaptacyjnych węzła najbardziej podobnego. Celem optymalizacji jest taka zmiana parametrów węzła, aby zwiększyć pobudzenia tego węzła dla aktualnego wektora wejściowego. Dokonuje się to poprzez zmianę położenia oraz rozmycia węzła podobnie jak w sieci RAN

$$D_i = D_i + \frac{\gamma}{m}\left(x_i - D_i\right) \tag{5.75}$$

$$\sigma_i = \sigma_i + \kappa\frac{(1-G(\mathbf{x};\mathbf{D},\sigma))\,|\,x_i - D_i\,|}{1+\dfrac{\tau-\tau_n}{\Lambda}} \tag{5.76}$$

gdzie parametry γ i κ mają następującą postać

$$\gamma = \frac{\Gamma}{1+\dfrac{\tau-\tau_n}{\Lambda}} \tag{5.77}$$

$$\kappa = \frac{\mathrm{K}}{1+\dfrac{\tau-\tau_n}{\Lambda}} \tag{5.78}$$

Parametr Λ określa szybkość zmniejszania się parametrów uczenia (parametr ten ustalony jest na początku uczenia i właściwie we wszystkich dotychczasowych zastosowaniach miał zawsze taką samą wartość, 100), natomiast Γ i K są początkowymi wartościami parametrów uczenia, które trzeba ustalić przed rozpoczęciem uczenia.

Oprócz zmiany rozmycia i położenia następuje zwiększenie masy węzła

$$m = m + 1 \tag{5.79}$$

Po zakończeniu każdej epoki masa węzła jest zerowana.

W celu uniknięcia zbyt silnego nakładania się węzłów reprezentujących różne klasy na siebie, w czasie optymalizacji podejmowana jest również próba zmniejszenia podobieństwa do węzła będącego największym zagrożeniem.

Po wykonaniu optymalizacji węzeł zostaje wyłączony z sieci. Jeżeli dla pozostałych węzłów $S_2(\mathbf{x}^i) = 0$, to następuje zmniejszenie rozmycia węzła najbardziej wzbudzającego się

$$\sigma_i = \left|\sigma_i - \xi G(\mathbf{x};\mathbf{D},\sigma)\frac{|\,x_i - D_i\,|}{1+\dfrac{\tau-\tau_n}{\Lambda}}\right| \tag{5.80}$$

gdzie

$$\xi = \begin{cases} 1.5 : G(\mathbf{x};\mathbf{D},\sigma) > 0.6 \\ 1 : w\ przeciwnym\ razie \end{cases} \tag{5.81}$$

Oprócz zmniejszenia rozmycia następuje odmłodzenie neuronu

$$\tau_n = \tau_n + \frac{\tau - \tau_n}{2} \tag{5.82}$$

Sieć FSM łączy ze sobą pewne cechy występujące w sieci RAN oraz RecBFN, gdyż występuje tu zarówno rozrastanie się węzłów jak i kurczenie wtedy, gdy jest to konieczne. W przeciwieństwie do sieci RecBFN nowym węzłom nie nadaje się od początku rozmyć tak dużych jak to tylko możliwe, lecz pozwala się, aby węzły od możliwych do przyjęcia w chwili początkowej rozmyć powiększały się na zasadzie konkurencji, powiększeniu ulega tylko zwycięzca.

Ponieważ w sieci FSM stosuje się różne funkcje transferu w przypadku niektórych z niech występuje konieczność rozbudowania algorytmu uczenia o dodatkowe parametry, lub też dokonania drobnych zmian w adaptacji parametrów przedstawionych powyżej. Powyższe wzory dotyczą tylko funkcji gaussowskiej i trapezoidalnej w przypadku pozostałych funkcji proces uczenia przebiega w następujący sposób:

1. Funkcje z obrotami wymagają zmiany katów obrotu w trakcji uczenia. Zmiany te dokonywane są dla każdego wektora wejściowego w tym samym momencie co optymalizacja wszystkich pozostałych parametrów, zgodnie z następującą relacją

$$\phi_i = \phi_i + \gamma(\alpha_i - \phi_i) \tag{5.83}$$

   ϕ aktualne kąty obrotu dla danego neuronu, α kąty reprezentujące dany wektor wejściowy, γ parametr uczenia zdefiniowany we wzorze (5.77).

   Każdy nowo utworzony neuron ma ustawione kąty obrotu na 0, ponieważ obejmuje tylko jeden wektor wejściowy.

2. Funkcje prostokątne mają tylko dwa możliwe wartości wzbudzeń 1 i 0. Występuje więc konieczność modyfikacji relacji (5.76) gdyż w przypadku wzbudzania się węzła przy wejściu $\mathbf{x}$ nie nastąpi jego poszerzenie. Zmieniona relacja (5.76) ma postać

$$\sigma_i = \sigma_i + \frac{\kappa}{2} \frac{|x_i - D_i|}{1 + \frac{\tau - \tau_n}{\Lambda}} \tag{5.84}$$

   natomiast relacja (5.80)

$$\sigma_i = \left| \sigma_i - \frac{3}{4} \xi \frac{|x_i - D_i|}{1 + \frac{\tau - \tau_n}{\Lambda}} \right| \tag{5.85}$$

   W funkcji prostokątnej niesymetrycznej (5.57) występują dwa niezależne rozmycia dla każdego wymiaru. Powiększanie obu rozmyć jest takie samo i dane jest relacją (5.84). Zmniejszanie natomiast jest zależne od tego, po której stronie położenia węzła w danym wymiarze wektor wejściowy się znajduje i tylko to rozmycie jest zmniejszane

$$\sigma_{i1} = \left| \sigma_{i1} - \frac{3}{4}\xi \frac{D_i - x_i}{1 + \frac{\tau - \tau_n}{\Lambda}} \right| gdy\, D_i - x_i > 0$$
$$\sigma_{i2} = \left| \sigma_{i2} - \frac{3}{4}\xi \frac{x_i - D_i}{1 + \frac{\tau - \tau_n}{\Lambda}} \right| gdy\, x_i - D_i > 0 \qquad (5.86)$$

3. Dla funkcji trapezoidalnej uczenia przebiega tak samo jak dla funkcji prostokątnej wtedy, gdy wektor wejściowy znajduje się w obszarze prostokątnym, lub tak jak dla funkcji trójkątnej, gdy wektor znajduje się w obszarze trójkąta tej funkcji.

8.5.4. Selekcja cech

W sieciach z funkcjami zlokalizowanymi i separowalnymi cechy zbędne są to takie cechy, których dowolnie duże rozmycie nie powoduje zmian w dokładności klasyfikacji danego węzła. W sieci RecBFN selekcja cech wbudowana jest w algorytm uczenia, gdyż każdy nowo utworzony węzeł od początku dla aktualnie nie istotnych wymiarów ma rozmycia nieskończenie duże, które w miarę dodawania nowych węzłów mogą ulec zmniejszeniu.

W sieci FSM natomiast selekcja cech wykonywana jest po zakończeniu procesu uczenia. Selekcja wykonywana jest dla każdego węzła, istniejącego w sieci, osobno. W celu przyspieszenia procesu selekcji dla każdego z węzłów odrzucane są z ciągu treningowego wektory, które są przez niego klasyfikowane. Wyłączenie cechy w danym węźle, z uwagi na to, że stosowana są funkcje separowalne, może być traktowane jak wstawienie do iloczynu wartości 1, a więc maksymalnej wartości aktywacji dla danego wymiaru. Może to spowodować tylko zwiększenie aktywacji węzła dla wszystkich wektorów treningowych. Nie może się więc zmienić klasyfikacja wektorów, które były poprawnie klasyfikowane, bo dla tych wektorów ten węzeł i tak miał największe wzbudzenie. Dlatego obliczamy macierz wzbudzeń tylko dla pozostałych wektorów treningowych. Każda z cech dla danego węzła zostaje kolejno odrzucana. Powoduje to wzrost funkcji aktywacji tego węzła. Jeśli aktywacja ta dla wektorów z innej klasy niż optymalizowany węzeł nie przekracza aktywacji węzłów poprawnie je klasyfikujących, zapisanych w tablicy aktywacji, to taka cecha nie jest potrzebna.

Inną metodą selekcji cech jest podobnie jak w przypadku sieci MLP-LN jest dodanie do funkcji błędu członu karzącego za małe wartości rozmyć

$$E = E_0 + \lambda \sum_k \sum_i \frac{1}{1 + \sigma_{ki}} \qquad (5.87)$$

gdzie E_0 jest funkcją błędu, λ parametr określający wpływ członu kary na funkcję błędu, k indeks przebiegający po węzłach ukrytych, i indeks przebiegający po poszczególnych wymiarach. Funkcja te w aktualnej wersji sieci FSM nie była stosowana gdyż istniejący algorytm uczenia jest algorytmem *ad-hoc*. W przypadku zastosowania średniokwadratowej funkcji błędu i minimalizacji gradientowej selekcja cech poprzez zastosowanie relacji (5.87) wydaje się być dobrym podejściem.

Po wykonaniu selekcji cech następuje odrzucanie węzłów, których odrzucenie nie powoduje zmniejszenia jakości klasyfikacji sieci, oraz węzłów małych. Następnie wykonywana jest ponownie selekcja cech i odrzucanie. Cała procedura powtarzana jest tak długa, aż odrzucanie węzłów nie będzie już możliwe.

8.5.5. Rozpoznawanie

Testowanie sieci FSM polega na wyszukiwaniu dla danego wektora wejściowego węzła, który ulega największemu wzbudzeniu. Wektor wejściowy jest przypisany do klasy, którą reprezentuje zwycięski neuron. Jeżeli jednak żaden z węzłów nie ulega wzbudzeniu, wówczas wektor wejściowy nie zostaje sklasyfikowany.

Do oszacowania jakości klasyfikacji służą dwa parametry: wartość wzbudzenia oraz współczynnik zaufania.

Wartość wzbudzania G jest z zakresu [0,1]; im jest większa tym wektor wejściowy bliższy jest istniejącemu prototypowi, który powstał podczas uczenia. Pomimo, że wartość wzbudzenia jest duża może się zdarzyć, że dużemu wzbudzeniu ulega jeszcze inny węzeł reprezentujący inną klasę. Dlatego też potrzebny jest dodatkowy parametr.

Współczynnik zaufania można traktować jako prawdopodobieństwo poprawnego zaklasyfikowania wektora wejściowego $\mathbf{x}$ do danej klasy C_i

$$p_{C_i}(\mathbf{x}) = \frac{\sum_k G(\mathbf{x}; D_k^{C_i}, \sigma_k^{C_i})}{\sum_j G(\mathbf{x}; D_j, \sigma_j)} \qquad \sum_i p_{C_i}(\mathbf{x}) = 1 \tag{5.88}$$

W liczniku sumowanie przebiega po wszystkich węzłach reprezentujących klasę C_i, w mianowniku suma biegnie po wszystkich węzłach ukrytych w sieci.

Do dokonania oceny klasyfikacji potrzebne są obydwa parametry, gdyż w przypadku niewielkiego wzbudzenia neuronu zwycięskiego, gdy nie wzbudzają się inne węzły, wartość współczynnika zaufania jest duża natomiast ze względu na nie duże wzbudzenie klasyfikacja może być wątpliwa. Podobnie w przypadku dużego wzbudzenia jednego węzła może się zdarzyć, że dużemu wzbudzeniu ulega jeszcze inny węzeł reprezentujący inną klasę w wyniku czego wzbudzenie jest duże natomiast współczynnik zaufania będzie mały.

8.5.6. Wady i zalety sieci FSM

Zalety:

1. Konstruktywistyczny algorytm uczenia. Chociaż opisany tu algorytm wydawać się może początkowo skomplikowany i trudny w użyciu w rzeczywistości wykonuje się niemal automatycznie, nie wymagając ingerencji użytkownika w proces uczenia się lub wybierania architektury.
2. Możliwość zastosowania różnych separowalnych funkcji transferu, również nieróżniczkowalnych

3. Możliwość wyciąganie reguł logicznych, które w wielu przypadkach dorównują algorytmom specjalizowanym do tego celu np. C4.5.
4. Możliwość pracy z wartościami brakującymi oraz ich uzupełniania.

Wady:

1. Brak stabilności w sensie określonym przez Breimana szczególnie dla problemów zawierających małą liczby danych treningowych. Poprawę stabilności w sieci FSM można uzyskać poprzez zastosowanie komitetu sieci. Jak pokazują rezultaty metoda ta dla sieci FSM działa bardzo dobrze.
2. Algorytm uczenia jest algorytmem *ad-hoc*.

9. Rezultaty

W rozdziale tym przedstawione są rezultaty działania opisanych algorytmów MLP2LN, oraz FSM. Algorytmy te porównane zostały na różnych bazach danych z najlepszymi klasyfikatorami, których wyniki dla danej bazy danych udało się znaleźć. Porównania uwzględniają też wyniki otrzymane za pomocą publicznie dostępnych programów innych autorów oraz programów opracowanych w Katedrze Metod Komputerowych (oznaczenie KMK). Dla każdej z baz danych są to na ogół inne klasyfikatory, ponieważ nie ma takiego klasyfikatora, który byłby najlepszy dla dowolnego problemu. Tezę tą potwierdzają wyniki otrzymane w projekcie STATLOG [40]

Algorytm	**Krótki opis**
CART	Binarne drzewo decyzyjne.
C4.5 [49]	Drzewo decyzyjne.
DIPOL92 [40]	Klasyfikator kawałkami liniowy, na pograniczu sieci neuronowej i statystycznych metod dyskryminacyjnych.
LogDisc	Dyskryminacja logistyczna.
SMART [40]	Analiza regresyjna.
LDA (Linear Discriminant Analysis)	Liniowa analiza dyskryminacyjna, znajduje prostą dyskryminacyjną poprzez maksymalizację prawdopodobieństwa.
FDA (Fisher Discriminant Analysis) [55]	Analiza dyskryminacyjna Fishera, znajduje prostą dyskryminacyjną poprzez maksymalizację separacji klas w sensie błędu średniokwadratowego.
PVM [67]	
IncNet [30]	Konstruktywistyczna sieć neuronowa, w której kontroluje się złożoność za pomocą filtru Kahlmana.
LVQ (Learning Vector Quantization) [34]	Sieć neuronowa, łącząca metody uczenia bez nadzoru z uczeniem pod nadzorem.
k-NN (*k*-Nearest Neighbour) [55]	*k*-najbliższych sąsiadów, klasyfikacja nieznanego obiektu odbywa się na podstawie najbliższych *k* sąsiadów ze zbioru treningowego.
Naïve Bayes (NB) [40]	Klasyfikator Bayesowski, opiera się na wyznaczeniu łącznego rozkładu klas i atrybutów na podstawie założenia niezależności poszczególnych atrybutów.
SNB-semi-naive Bayes (pairwise dependent) [59]	Zmodyfikowany NB.
AQ15	Uczenie indukcyjne.
ASI (Assistant-I) [24]	Drzewo decyzyjne, w którym łączy się wszystkie wartości cechy w zbiory dające największą wartość funkcji jakości podziału.

Tabela 9-1 Opis klasyfikatorów, z którymi porównane zostały algorytmy MLP2LN oraz FSM.

W niektórych danych przedstawionych w tym rozdziale występują wartości brakujące. W przypadku algorytmów oceniających pojedyncze cechy, takich jak C4.5, CART, PVM, wartości brakujące nie sprawiają większego problemu. Dla sieci neuronowych (MLP+backprop, MLP2LN) wartość brakująca została zastąpiona wartością średnią dla danej cechy. Przy użyciu sieci FSM wartość brakująca zazwyczaj traktowana jest jako wartość brakująca z omijaniem (8.2 Dane brakujące). W przypadku użycia sieci FSM dane przed przestąpieniem do uczenia zostały zestandaryzowane. Każdy typ danych została sklasyfikowany za pomocą sieci FSM przy użyciu różnych funkcji transferu oraz różnych metod. Poniższa tabela przedstawia spis skrótów używanych przy opisie wyników uzyskanych za pomocą sieci FSM.

Oznaczenie	Wyjaśnienie
FT	Trójkątna funkcja transferu
FG	Gaussowska funkcja transferu
FP	Prostokątna asymetryczna funkcja transferu, wykorzystywana do tworzenia reguł logicznych
R	Podczas uczenia funkcje aktywacji są obracane
CN	Komitet sieci, końcowa klasyfikacja jest rezultatem głosowania nauczonych sieci

Tabela 9-2 Spis oznaczeń występujących w opisie wyników sieci FSM.

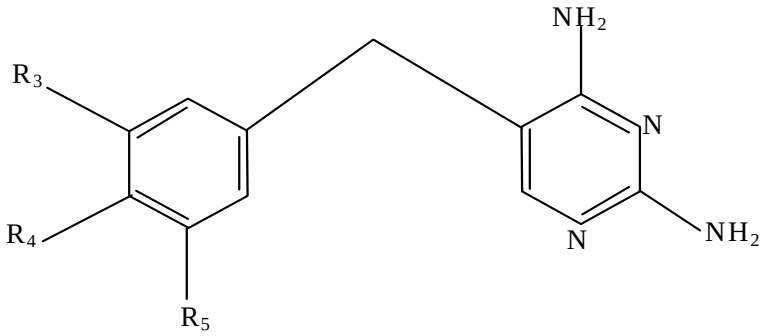
Niektóre opcje mogą być ze sobą łączone. W tabelach z wynikami przedstawione jest to za pomocą znaku „+” np. FG+R+CN=10 – zastosowano gaussowską funkcję transferu wraz z możliwością jej obrotu, oraz użyty został komitet sieci składający się z 10 sieci. Wszystkie rezultaty sieci FSM powstały z uśrednienia 10 wyników otrzymanych dla określonej bazy danych, wyliczone jest również odchylenie standardowe, które przedstawione jest obok wyniku uśrednionego. Przy ekstrakcji reguł za pomocą sieci FSM do danych niesklasyfikowanych przez wygenerowane reguły została zastosowana reguła *else*. Reguła ta przydziela niesklasyfikowane wektory do klasy, która została najgorzej opisana przez FSM tzn. występuje duża liczba reguł opisująca tą klasę lub też występuje duża liczba wektorów niesklasyfikowanych z ciągu treningowego dla tej klasy.

Przy użyciu algorytmy MLP2LN dokonana została wstępna dyskretyzacja danych. Najczęściej używana była metoda oparta na histogramach. Niestety sieć MLP2LN nie została użyta do wszystkich danych przedstawionych tutaj, gdyż jak do tej pory dobieranie parametrów i ich zmiana dokonywane były ręcznie. Jest to proces bardzo czasochłonny i dlatego nie wszystkie dane mogły zostać w ten sposób przeanalizowane.

9.1. Dane SAR (Structure Activity Relationships)

9.1.1. Pirymidiny

Dane pobrane zostały z UCI [6]. Wszystkie pirymidiny mają wspólny szablon przedstawiony poniżej



Rys 9:1 Szablon reprezentujący pirymidiny.

Pozycje R_3, R_4, R_5 są miejscami, w których mogą wystąpić różne związki chemiczne. Każda z tych pozycji opisana jest przez 9 cech: nazwa związku, polarność, dawca wiązania wodorowego, biorca wiązania wodorowego, dawca pi, biorca pi, polaryzowalność, efekt sigma. Każda pirymidina opisana jest więc za pomocą 27 cech. Niekiedy na dwóch lub jednej pozycji podstawieniowej nie występuje żaden związek chemiczny. W takiej sytuacji cechy opisujące daną pozycję oznaczone są jako brakujące. W tym jednak wypadku brak podstawienia nie jest wynikiem braku informacji w związku z tym cechy opisujące te pozycje nie są traktowane jako brakujące. W miejsca te zostały wstawione wartości unikalne.

Celem jest rozpoznanie, który z dwóch związków ma większą aktywność. Dlatego też każdy wektor w bazie danych składa się z dwóch pirymidin. W związku z tym wektory treningowe składają się z 54 cech. Występują dwie klasy pierwsza z nich oznacza, że pierwszy związek w parze ma większą aktywność, druga odwrotnie. Dane zostały podzielone na pięć zbiorów treningowych i testowych. W poniższej tabeli przedstawiono wyniki kilku klasyfikatorów, które otrzymane zostały z uśrednienia wyników ze wszystkich pięciu zbiorów danych.

Algorytm	Współczynnik Spearmana
	5CV
Golem (ILP)	0.684
Regresja liniowa	0.654
CART	0.499

Tabela 9-3 Porównanie kilku klasyfikatorów dla danych opisujących pirymidiny.

Współczynnik Spearmana zdefiniowany jest w następujący sposób:

$$r_S = 1 - \frac{6}{n^3 - n} \sum_i d_i^2$$

gdzie d odległość w rankingu par (parą nazywamy wartość wyjściową dla aktualnego wektora wejściowego z klasyfikatora oraz rzeczywistą wartość dla tego wektora), n liczba par. Współczynnik ten ma wartość [-1,1], 1 oznacza doskonałą dodatnią korelacje i –1 doskonałą ujemną korelacje.

Algorytm Golem opisuje dane za pomocą 9 prologowych klauzul, w przypadku zastosowania metody regresyjnej otrzymuje się równanie zawierające 7 niezależnych

zmiennych. Drzewo CART natomiast opisuje dane za pomomocą 49 reguł. Wyniki otrzymane za pomocą sieci FSM są zacznie lepsze od przedstawionych w tabeli 9-4. Zarówno próba wyciągania reguł jak i opis danych za pomocą funkcji gaussowskich daje ten sam wynik. FSM generuje średnio około 41 reguł opisujących obie klasy. Jest to, co prawda znacznie bardziej skomplikowany opis niż ten otrzymywany przez algorytm Golem, ale generalizacja sieci FSM jest dużo lepsza.

Algorytm	Jakość klasyfikacji [%] Na zbiorze testowym	Średnia liczba węzłów
FG	0.77+/-0.018	86
FP	0.77+/-0.033	41

Tabela 9-4 Wyniki otrzymane przy użyciu sieci FSM z różnymi funkcjami transferu dla danych reprezentujących pirymidiny.

Obliczenia przedstawione w tabeli 9-4 wykonane zostały tylko dla dwóch funkcji transferu.

9.1.2. PTE (The predictive toxicology evaluation)

Dane zostały wzięte z adresu http://oldwww.comlab.ox.ac.uk/oucl/groups/machlearn/PTE. Występuje 330 związków organicznych, które zostały zebrane w raporcie NTP (US National Toxicology Program). Z pośród nich 182 (55%) zaklasyfikowano jako rakotwórcze, 148 nie rakotwórcze. Każdy z wektorów treningowych opisany jest za pomocą 417 cech. Celem jest uzyskanie teorii i użycie jej do opisu 30 testowych wektorów (PTE-2). Spośród tych 30 wektorów tylko dla 20 znana jest rakotwórczość. Występuje tutaj bardzo wiele brakujących wartości, wszystkich wartości brakujących jest 6323. Przy użyciu sieci FSM wartości brakujące podczas obliczeń traktowane są poprzez pomijanie odpowiednich wymiarów.

Algorytm	Jakość klasyfikacji [%] Na zbiorze testowym	Źródło wyniku
Distill-Light	87	machlearn/PTE/pte2-summary.html
STEPS (algorytm stochastyczny)	78	machlearn/PTE/pte2-summary.html
GloBo	78	machlearn/PTE/pte2-summary.html
OFAI	74	machlearn/PTE/pte2-summary.html
Default	70	

Tabela 9-5 Porównanie kilku klasyfikatorów dla danych PTE.

Wyniki przedstawione w tabeli 9-5 prezentują rozwiązania dla danych testowych, w których spośród 30 wektorów w 23 znana jest rakotwórczość. Wyniki przedstawione w poniższej tabeli uzyskane za pomocą FSM uzyskane zostały na danych, w których tylko dla 20 wektorów znana jest rakotwórczość. Nie udało się niestety zdobyć informacji o 3 dodatkowych wektorach.

Powyższy rezultat sieci FSM jest najlepszym, jaki udało się osiągnąć po wielu próbach uczenia.

9.2. Segmentacja obrazu

Dane zostały pobrane z projektu STATLOG [40]. Zostały utworzone poprzez losowy wybór z bazy danych 7 kolorowych zdjęć. Zdjęcia te zostały ręcznie podzielone na części tak, aby zaklasyfikować każdy z pikseli do jednej z klas: cegła, niebo, listowie, cement, okno, droga, trawa. W ciągu treningowym znajduje się 2310 wektorów opisanych za pomocą 19 cech. Każda z klas reprezentowana jest przez 330 wektorów.

Algorytm	Jakość klasyfikacji [%]	Źródło wyniku
	10CV	
Alloc80	97.0	[40]
AC^2	96.9	[40]
Baytree	96.7	[40]
NewID	96.6	[40]
DIPOL92	96.1	[40]
C4.5	96.0	[40]
Default	14.3	

Tabela 9-6 Porównanie wyników różnych klasyfikatorów dla danych segmentacji obrazu.

Wyniki uzyskane za pomocą sieci FSM są bardzo zbliżone do wyników najlepszych klasyfikatorów opisanych w tabeli 9-6. Zastosowanie funkcji prostokątnej wraz z komitetem 20 sieci daje wynik lepszy od otrzymanego za pomocą algorytmu Alloc80. Niestety obliczenia wykonywane za pomocą sieci FSM są bardzo czasochłonne. Powodem jest bardzo duży zbiór danych oraz to, że wykonanie 10CV wraz z komitetem 20 sieci i 10-krotne powtórzenie wymaga nauczenia 2000 sieci. Przy założeniu, że uczenie jednej sieci trwa 1 minutę nauczenie 2000 sieci trwa 1.3 dnia.

Algorytm	Jakość klasyfikacji [%] 10CV	Średnia liczba węzłów
FT	95.91+/-0.23	58
FT+CN=20		
FG	96.37+/-0.27	58
FG+CN=20	96.95+/-0.15	58
FG+R	96.12+/-0.19	60
FG+R+CN=20	96.3+/-0.27	60
FP	95.85+/-0.33	31
FP+CN=20	97.6+/-0.18	31

Tabela 9-7 Wyniki otrzymane przy użyciu sieci FSM z różnymi funkcjami transferu dla danych segmentacji obrazu.

9.3. Dane „Shuttle”

Dane zostały pobrane z projektu STATLOG [40]. Oryginalne dane pochodzą z NASA i dotyczą położenia radiatorów wewnątrz wahadłowca. Dane zostały podzielone na zbiór treningowy zawierający 43500 wektorów oraz zbiór testowy 14500 wektorów. W zbiorze trenin-

gowym występuje 6 klas zawierających następującą liczbę wektorów: 34108 (78.4%), 37 (0.08%), 132 (0.3%), 6748 (15.5%), 2458 (5.6%), 17 (0.04%). W zbiorze testowym liczba wektorów dla poszczególnych klas jest następująca: 11478 (79.2%), 13 (0.0897%), 39 (0.269%), 2155 (14.9%), 809 (5.58%), 4 (0.0276%), 2 (0.0138%). W zbiorze testowym występują dwa wektory z klasy, która nie istnieje w zbiorze treningowym. Każdy z wektorów znajdujących się w tej bazie danych opisany jest za pomocą 9 cech.

Algorytm	Jakość klasyfikacji [%] Na zbiorze testowym	Źródło wyniku
NewID	99.99	[40]
Baytree	99.98	[40]
Cal5,CN2	99.97	[40]
CART	99.92	[40]
C4.5	99.9	[40]
Default	79.2	

Tabela 9-8 Porównanie wyników różnych klasyfikatorów dla danych „Shuttle".

Najlepsze rezultaty uzyskiwane są przez klasyfikatory regułowe NewID, CN2, CART, C4.5. W sieci FSM zastosowanie dowolnej funkcji aktywacji daje podobne wyniki. Najlepszy rezultat otrzymywany jest przy zastosowaniu gaussowskiej funkcji aktywacji i komitetu sieci.

Algorytm	Jakość klasyfikacji [%] Na zbiorze testowym	Średnia liczba węzłów
FT	99.91+/-0.02	32
FT+CN=20		
FG	99.92+/-0.03	36
FG+CN=20	99.94+/0.008	36
FG+R	99.9+/-0.02	39
FG+R+CN=20	99.93+/-0.01	39
FP	99.91+/-0.03	16
FP+CN	99.91+/-0.06	16

Tabela 9-9 Wyniki otrzymane przy użyciu sieci FSM z różnymi funkcjami transferu dla danych „Shuttle".

9.4. DNA

Dane zostały pobrane z projektu STATLOG [40]. Podczas tworzenia protein zbędna część DNA zostaje usunięta. Punkty w sekwencji DNA, w których następuje to usunięcie nazywane są punktami sklejenia. Celem jest rozpoznanie, przy zadanej sekwencji DNA, granic pomiędzy częścią DNA, która pozostaje po usunięciu - extron, a częścią, która zostaje usunięta - intron. Każdy z 3186 wektorów występujących w bazie składa się z okna 60 nukleotydów, które reprezentują jedną z 4 symbolicznych wartości (A,C,G,T). Cechy zostały zakodowane w następujący sposób: A – 1 0 0, C – 0 1 0, G – 0 1 0, T – 0 0 0, stąd liczba cech opisująca każdy wektor treningowy wynosi 180. Klasa wyznaczona jest dla środkowego punktu w oknie. Występują trzy klasy: extron-intron, intron-extron, lub też ani jeden ani drugi. Dane zostały

podzielone na zbiór treningowy, w którym występuje 2000 wektorów oraz zbiór testowy 1186 wektorów.

Algorytm	Jakość klasyfikacji [%] Na zbiorze testowym	Źródło wyniku
RBF	95.9	[40]
DIPOL92	95.2	[40]
ALLOC80	94.3	[40]
Quadisc	94.1	[40]
Logdisc	93.9	[40]
NaiveBay	93.2	[40]
Default	50.8	

Tabela 9-10 Porównanie wyników różnych klasyfikatorów dla danych „DNA".

Najlepszy wynik dla sieci FSM otrzymuje się przy użyciu funkcji gaussowskiej oraz komitetu 20 sieci. Wynik ten jest taki sam jak wyniki algorytmu Alloc80. Funkcja prostokątna w przypadku tych danych nie została zastosowana, ponieważ przy użyciu tej funkcji nie udało się nauczyć sieci FSM do rozsądnego poziomu.

Algorytm	Jakość klasyfikacji [%] Na zbiorze testowym	Średnia liczba węzłów
FT	93.6+/-0.4	5
FT+CN=20	94+/-0.37	5
FG	94.1+/-0.38	7
FG+CN=20	94.3+/-0.18	7
FG+R	94.1+/-0.44	7
FG+R+CN=20		

Tabela 9-11 Wyniki otrzymane przy użyciu sieci FSM z różnymi funkcjami transferu dla danych „DNA".

9.5. Dane „sonar"

Dane zostały zebrane przez Gormana i Sejnowskiego podczas pracy nad klasyfikacją danych sonarowych za pomocą sieci neuronowych. Zostały pobrane z UCI [6]. Celem klasyfikacji danych jest odróżnienie sygnału sonarowego, powstałego w wyniku odbicia od cylindrycznej powierzchni metalicznej, od odbicia od skały mającej kształt w przybliżeniu cylindryczny. Wektory opisane są za pomocą 60 cech. Występuje zbiór treningowy składający się ze 104 wektorów, w którym znajduje się 49 wektorów z jednej klasy, 55 wektorów z drugiej klasy, oraz zbiór testowy składający się ze również ze 104 wektorów w tym 64 z jednej i 42 wektory z drugiej klasy.

Algorytm	Jakość klasyfikacji [%] Na zbiorze testowym	Źródło wyniku
TAP MFT Bayesian	92.3	[44]
MLP+BP	90.4	[44]

IBL	87.02	[68]
Default	40.4	

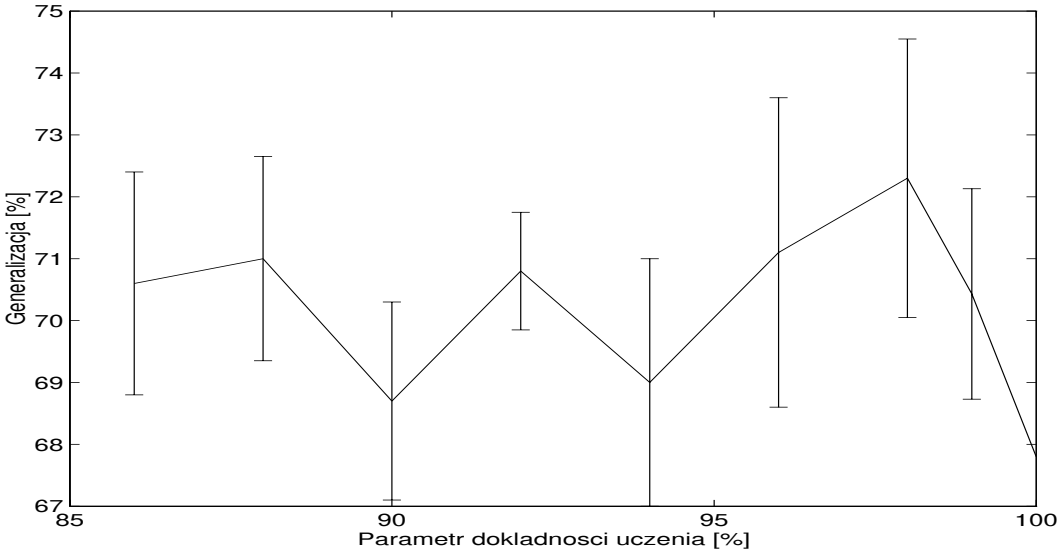
Tabela 9-12 Porównanie wyników różnych klasyfikatorów dla danych „sonar”.

Dane te były analizowane przez dużą większą liczbę systemów niż wymienione w tabeli 9-12. Niestety większość badaczy stosowała swoje własne metody wyznaczania generalizacji np. 10CV (pomimo, że istnieje wydzielony zbiór testowy). W przypadku sieci FSM wykorzystany został zbiór testowy.

Algorytm	Jakość klasyfikacji [%] Na zbiorze testowym	Średnia liczba węzłów
FT	79.2+/-6.0	27
FT+CN=40	82.2+/-3.0	27
FG	85+/-3.0	9
FG+CN=40	89.7+/-2.0	9
FG+R	83.7+/-3.0	10
FG+R+CN=40	90.9+/-2.0	10
FP	73+/-6.0	7
FP+CN=40	83.7+/-4.0	7

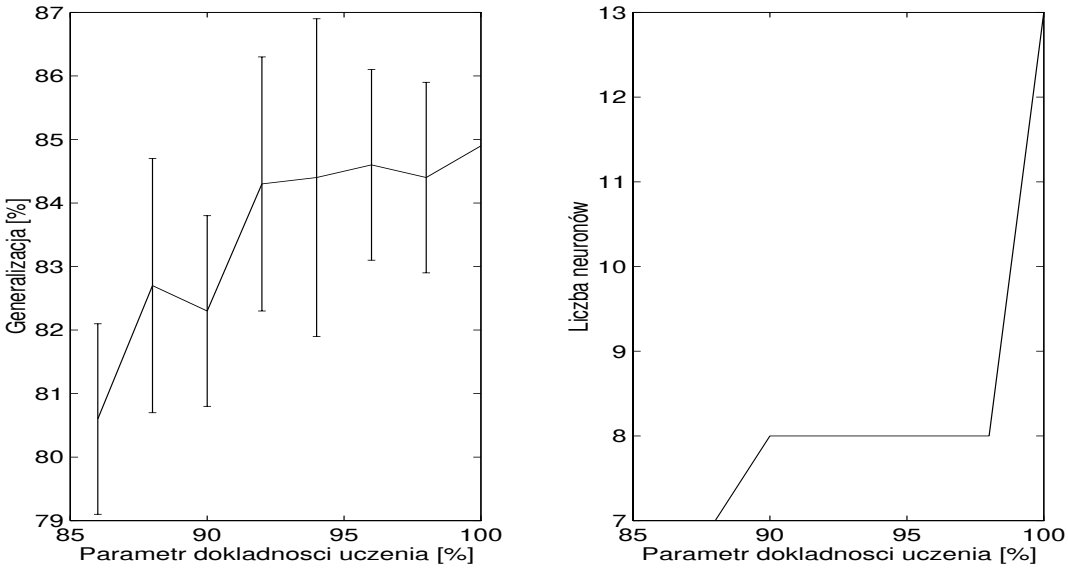
Tabela 9-13 Wyniki otrzymane przy użyciu sieci FSM z różnymi funkcjami transferu dla danych sonar.

Parametr uczenia dla sieci FSM został wyznaczony na podstawie wyników otrzymanych z testu 10CV na zbiorze treningowym. Poniższy rysunek przedstawia zależność generalizacji od parametru uczenia dla funkcji gaussowskiej w teście CV. Dla każdego parametru uczenia test 10CV powtórzony został 10 razy, wyniki zostały uśrednione. Dla poszczególnych wartości parametru uczenia nie widać dużego zróżnicowania w generalizacji, dopiero dla parametru uczenia 99% i 100% nastąpiło zmniejszenie generalizacji. Z tego powodu, do sprawdzenia generalizacji na zbiorze testowym, parametr uczenia sieci FSM ustalony został na 98%.



Rys 9:2 Zależność dokładności klasyfikacji sieci FSM, dla danych „sonar" w teście 10CV, od parametru dokładności uczenia stosowanego w FSM, wraz z zaznaczonym odchyleniem standardowym dla poszczególnych wyników.

Poniższy wykres przedstawia jak zmienia się jakość generalizacji (dokładność klasyfikacji na zbiorze testowym) od parametru uczenia sieci FSM przy zastosowaniu funkcji gaussowskiej (FG), oraz jak zmienia się liczba węzłów wraz ze wzrostem parametru uczenia. Najlepsza generalizacja uzyskiwana jest dla parametru uczenia 100% tzn. żądamy, aby na zbiorze treningowym dokładność uczenia była 100%. W takim momencie uczenie sieci jest przerywane i następuje odrzucanie węzłów, które zostają uznane jako małe. Tak, więc po zakończeniu sieć nie musi i nie ma dokładności 100%.



Rys 9:3 Lewy rysunek przedstawia zależność dokładności klasyfikacji sieci FSM, dla danych „sonar" na zbiorze testowym, od parametru dokładności uczenia stosowanego w FSM. Prawy rysunek reprezentuje zależność liczby węzłów od parametru dokładności uczenia.

Porównując wyniki otrzymane z testu 10CV na zbiorze treningowym z wynikami uzyskanymi na zbiorze testowym można zauważyć, że występują tylko drobne różnice. W wyniku testu 10CV najlepsza generalizacja jest dla parametru uczenia 98%, wartość generalizacji wynosi około 72%. Na zbiorze testowym najlepsza generalizacja uzyskiwana jest dla parametru uczenia 100%, natomiast jakość generalizacji jest znacznie większa około 85%.

Dane „galaxies” [36], zawierają opis galaktyk z katalogu ESO-LV. Celem jest wykonanie automatycznej klasyfikacji, która porównywalna jest do klasyfikacji eksperta. Zbiór ten zawiera ponad 5000 wektorów. Sieć trenowana jest na zbiorze zawierającym 1700 elementów i testowana na zbiorze 3517 wektorów. Każdy wektor składa się z 13 cech, które zostały opisane w poniższej tabeli.

1. średnia różnic koloru niebieskiego i czerwonego	2. eksponent w uogólnionym prawie Vacouleura dla koloru niebieskiego
3. logarytm ze stosunku średnicy zawierającej 80% koloru niebieskiego do średnicy zawierającej 50% koloru niebieskiego	4.wskaźnik stopnia asymetrii obrazu galaktyki
5. jasność centrum powierzchni dla koloru niebieskiego	6. logarytm ze stosunku średnicy mniejszej do większej
7. błąd w fitowaniu elipsy do izofoty dla koloru niebieskiego	8. gradient jasności powierzchni dla koloru niebieskiego
9. logarytm ze stosunku średnicy dla koloru niebieskiego przy 26 mag/arsec^2 do średnicy dla połowy światła	10. eksponent w uogólnionym prawie Vacouleura dla koloru czerwonego
11. średnia jasność dla powierzchni niebieskiej wewnątrz średnicy 10arsec przy okrągłej szczelinie	12. jasność powierzchni niebieskiej
13. jasność powierzchni czerwonej	

Tabela 9-14 Opis cech dla obiektów reprezentujących galaktyki.

Wszystkie obiekty w tej bazie danych podzielone zostały na dwa rodzaje galaktyk: wczesnego typu (424 wektory), lub też późnego typu (1276 wektorów). W zbiorze testowym występuje 889 obiektów opisujących galaktyki wczesnego typu i 2628 późnego typu.

Algorytm	Jakość klasyfikacji [%] Na zbiorze testowym	Źródło wyniku
k-NN	92.82	KMK
C4.5	92.4	KMK
MLP+BP	89.6	[36]
Default	74.7	

Tabela 9-15 Porównanie wyników różnych klasyfikatorów dla danych „typy galaktyk”.

Najlepszy wynik otrzymywany jest za pomocą algorytmu *k*-NN oraz C4.5, różnica wyników między tymi algorytmami jest niewielka. Podobne rezultaty otrzymuje się z sieci FSM. Najlepszy wynik uzyskany został przy zastosowaniu gaussowskiej funkcji transferu wraz z możliwością obrotu i komitetem 30 sieci. Bardzo dobry wynik uzyskiwany jest również przy wyciąganiu reguł. Powstaje tylko 14 reguł dających 91.2% dokładności na zbiorze testowym. Liczba węzłów tworzonych przez sieć FSM przy użyciu pozostałych funkcji jest znacznie większa, natomiast różnice w generalizacji wynoszą tylko około 1%.

Algorytm	Jakość klasyfikacji [%] Na zbiorze testowym	Średnia liczba węzłów
FT	92+/-0.2	66
FT+CN=30	92+/-0.07	66
FG	92.2+/-0.2	76
FG+CN=30	92.4+/-0.12	76
FG+R	92.2+/-0.22	71
FG+R+CN=30	92.6+/-0.1	71
FP	91.2+/-0.28	14
FP+CN=30	91.4+/-0.14	14

Tabela 9-16 Wyniki otrzymane przy użyciu sieci FSM z różnymi funkcjami transferu dla danych „typy galaktyk".

9.7. Dane „jonosfera"

Dane zostały pobrane z UCI [6]. Zawierają 351 wektorów, z czego pierwszych 200 używa się do trenowania, natomiast pozostałe do testowania. Dane te reprezentują sygnał radarowy odbity od jonosfery. Każdy z wektorów zawiera 34 atrybuty o wartościach ciągłych. Występują dwie klasy, pierwsza świadcząca o występowaniu w jonosferze pewnych struktur i druga, świadcząca o braku tych struktur. Obie klasy w zbiorze treningowym są prawie równoliczne.

Algorytm	Jakość klasyfikacji [%] Na zbiorze testowym	Źródło wyniku
3-NN	96.6	KMK
MLP+BP	96	[6]
Default	82	

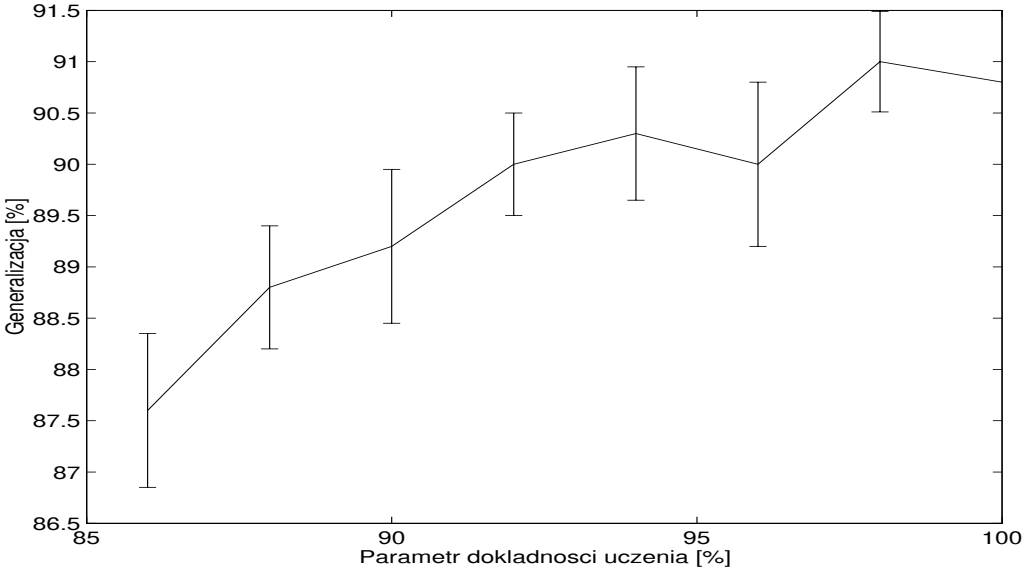
Tabela 9-17 Porównanie wyników różnych klasyfikatorów dla danych „jonosfera".

Algorytm	Jakość klasyfikacji [%] Na zbiorze testowym	Średnia liczba węzłów
FT	95.1+/-1.7	24
FT+CN=30	95.4+/-0.7	24
FG	96.2+/-1	19
FG+CN=30	97.8+/-0.3	19
FG+R	95.5+/-1	19
FG+R+CN=30	97.5+/-0.8	19
FP	88.1+/-6	10
FP+CN=30	94.6+/-0.5	10

Tabela 9-18 Wyniki otrzymane przy użyciu sieci FSM z różnymi funkcjami transferu dla danych „jonosfera".

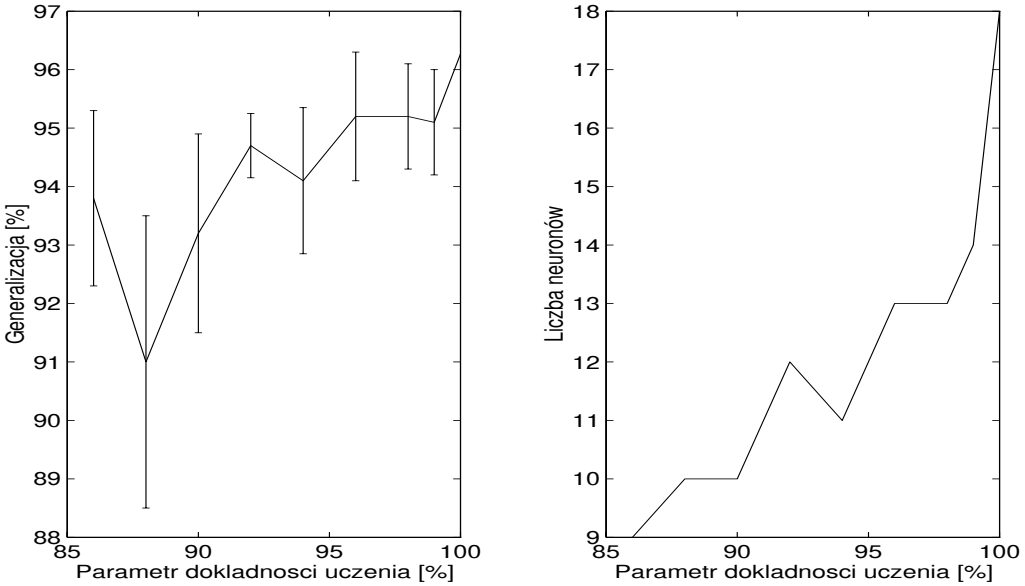
Podobnie jak dla danych „sonar" parametr uczenia sieci FSM wyznaczony został na podstawie testu 10CV na zbiorze treningowym. Zależność generalizacji podczas testu 10CV,

dla gaussowskiej funkcji aktywacji, od parametru uczenia przedstawia poniższy rysunek. Jak wynika z poniższego wykresu generalizacja rośnie wraz ze wzrostem parametru uczenia. Nawet przy maksymalnej wartości tego parametru (100%) wartość generalizacji nie spada znacząco. W związku z tym parametr uczenia sieci FSM, przy sprawdzaniu generalizacji na zbiorze testowym, ustalony został na wartość 100%.



Rys 9:4 Zależność dokładności klasyfikacji sieci FSM, dla danych „jonosfera" w teście 10CV, od parametru dokładności uczenia stosowanego w FSM, wraz z zaznaczonym odchyleniem standardowym dla poszczególnych wyników.

Poniższy rysunek przedstawia zależność generalizacji na zbiorze testowym dla danych „jonosfera" od parametru dokładności uczenia. Generalizacja jest tutaj największa dla parametru uczenia 100% podobnie jak w przypadku testu 10 CV. Oznacza to, że test 10CV doskonale nadaje się do wyznaczenia parametru uczenia sieci FSM.



Rys 9:2 Lewy rysunek przedstawia zależność dokładności klasyfikacji sieci FSM dla danych „jonosfera" od parametru dokładności uczenia stosowanego w FSM. Prawy rysunek reprezentuje zależność liczby węzłów od parametru dokładności uczenia.

9.8.1. Cukrzyca

Dane to zostały pobrane z UCI [6]. Powstały w wyniku wyselekcjonowania z dużej bazy danych National Institute of Diabetes and Digestive and Kidney Diseases. Wszyscy pacjenci występujący w tej bazie są płci żeńskiej w wieku co najmniej 21 lat, pochodzą z plemienia Pima (Arizona), stąd w literaturze baza ta znana jest jako „Pima Indian diabetes".

Celem jest dokonanie predykcji czy test na cukrzycę wykonany na danym pacjencie będzie pozytywny tzn. spełnione będą kryteria światowej organizacji zdrowia, przy zadanych fizjologicznych pomiarach i testach medycznych. Każdy przypadek opisany jest za pomocą 8 cech opisanych w poniższej tabeli.

N. cechy	Opis cechy	Liczba możliwych wartości
1	Liczba ciąż	Dyskretna
2	Test tolerancji glukozy	Ciągła
3	Ciśnienie rozkurczowe	Ciągła
4	Grubość zagięcia skóry	Ciągła
5	Poziom insuliny	Ciągła
6	Masa ciała	Ciągła
7	Funkcja rodowodowa cukrzycy	Ciągła
8	Wiek	Ciągła

Tabela 9-19 Opis cech dla danych cukrzycy.

500 przypadków należy do klasy z pozytywnym testem na cukrzycę i 268 do drugiej klasy. Nie występuje zbiór testowy, dlatego też do oszacowania generalizacji systemów uczących się zastosowano 10 krotną kroswalidację.

Algorytm	Jakość klasyfikacji [%]	Źródło wyniku
	10CV	
Logdisc	77.7	[40]
IncNet	77.6	[30]
DIPOL92	77.6	[40]
LDA	77.5-77.2	[40][59]
SMART	76.8	[40]
ASI	76.6	[59]
FDA	76.5	[59]
Default	65.1	

Tabela 9-20 Porównanie wyników różnych klasyfikatorów dla danych cukrzycy.

Na tej bazie danych bardzo dobre wyniki uzyskuje się stosując metody dyskryminacyjne. Aż 4 algorytmy dyskryminacyjne znajdują się w czołówce listy przedstawionej w tabeli 9-17 są to: Logdisc, DIPOL92, LDA, FDA. Wyniki otrzymywane za pomocą sieci FSM są gor-

sze o około 2%. Ponieważ zbiór testowy w każdym etapie CV składa się z około 77 wektorów, to błąd 2% oznacza błąd w klasyfikacji około 2 wektorów.

Algorytm	Jakość klasyfikacji [%] 10CV	Średnia liczba węzłów
FT	75.2+/-0.54	32
FT+CN=20	75.6+/-0.5	32
FG	74.8+/-0.73	30
FG+CN=20	75.18+/-0.55	30
FG+R	75.2+/-0.54	32
FG+R+CN=20	75.2+/-0.5	32
FP	73.5+/-0.93	13
FP+CN	74.3+/-0.32	13

Tabela 9-21 Wyniki otrzymane przy użyciu sieci dla danych cukrzycy.

9.8.2. Dane chorÛb tarczycy

Dane zostały pobrane z UCI [6].Występują trzy klasy zdrowy, nadczynność tarczycy, niedoczynność tarczycy. Poszczególni pacjenci opisani są przez 21 cech, z czego 15 jest binarnych i 6 ciągłych.

1. wiek: ciągła	2. płeć: binarna	3. leczenie tyroksyną: binarna
4. być może leczono tyroksyną: binarna	5. leki przeciwtarczycowe: binarna	6. samopoczucie pacjenta: binarna
7. czy pacjentka była w ciąży: binarna	8. czy wykonywano operacje tarczycy: binarna	9. czy leczono jodem J131 : binarna
10. test na niedoczynność tarczycy: binarna	11. test na nadczynność: binarna	12. czy jest stosowany cytrynian litowy: binarna
13. występowanie wola: binarna	14.występowanie nowotworu: binarna	15. czy występuje niedoczynność przysadki mózgowej: binarna
16. symptomy psychologiczne: binarna	17. poziom TSH: ciągła	18. poziom T3: ciągła
19. poziom TT4: ciągła	20. poziom T4U: ciągła	21. poziom FTI: ciągła

Tabela 9-22 Spis cech opisujących przypadki tarczycy.

W zbiorze treningowym występuje 3772 wektory, przy czym większość 3488 (92.4%) stanowią pacjenci zdrowi, nadczynność tarczycy reprezentowana jest przez 93 (2.5%) wektory, niedoczynność przez 191 (5.1%) wektorów. Stąd dobry klasyfikator powinien klasyfikować znacznie powyżej 92%. Występuje również zbiór testowy, w którym znajduje się 3428 wektorów, z czego 3178 (92.7%) reprezentuje pacjentów zdrowych, 73 (2.1%) z nadczynnością i 177 (5.2%) z niedoczynnością.

Najlepsze rezultaty dla tej bazy danych uzyskiwane są przez klasyfikatory regułowe takie jak CART czy PVM.. Bardzo dobry wynik uzyskany został przez IncNet, który jako jedyna

sieć neuronowa nie produkująca reguł uzyskuje tak dobry wynik. Niestety brak jest informacji o rozkładzie wyników uzyskiwanych przez sieć IncNet. W pozostałych przypadkach tzn. CART i PVM, wyniki są za każdym razem te same, więc informacja ta nie jest potrzebna.

Algorytm	Jakość klasyfikacji [%]		Źródło wyniku
	Zbiór treningowy	Zbiór testowy	
MLP2LN+ASA	99.9	99.36	KMK [17]
CART	99.8	99.36	[67]
PVM	99.8	99.36	[67]
IncNet	99.68	99.24	[30]
MLP2LN	99.70	99.00	KMK
Default	92%		

Tabela 9-23 Porównanie wyników różnych klasyfikatorów dla danych tarczycy.

MLP2LN jest algorytmem regułowym, wynik uzyskiwany przez tę metodę porównywalny jest z najlepszymi wynikam. Wyniki MLP2LN zostały ulepszone poprzez zastosowanie hybrydowej metody wyciągania reguł (MLP2LN+ASA [17]). Za pomocą algorytmu MLP2LN uzyskane zostały 4 reguły mające następującą postać

Nadczynność:

1. F17>29 i F21<63
2. F17=[6.1,29] i F21<63 i F18<20

Niedoczynność:

1. F17>6.1 i F21>[63,180] i F3=NIE i F8=NIE

Zdrowy:

1. Jeżeli nie powyższe reguły

Reguły te opisują poprawnie w 99.68% dane treningowe. Natomiast dane testowe w 99.07%.

Podobnie sieć FSM najlepsze wyniki daje z opcją wyciągania reguł. Reguły uzyskane z sieci FSM dotyczą wszystkich klas występujących w danych i w kolejnych próbach różnią się trochę od siebie. Dla przykładu zaprezentuje zbiór cech istotnych występujący w kilku różnych zestawach reguł:

- *F*1, *F*2, *F*3, *F*8, *F*17, *F*18, *F*19, *F*20, *F*21 liczba reguł 11 dokładność na zbiorze testowym 99.1
- *F*3, *F*8, *F*10, *F*12, *F*17, *F*18, *F*19, *F*21 liczba reguł 7 dokładność na zbiorze testowym 99.2
- *F*1, *F*2, *F*3, *F*8, *F*10, *F*17, *F*18, *F*19, *F*20, *F*21 liczba reguł 10 dokładność na zbiorze testowym 99.2
- *F*1, *F*3, *F*8, *F*10, *F*11, *F*17, *F*18, *F*19, *F*21 liczba reguł 9 dokładność na zbiorze testowym 99.2

Cechy, które powtarzają się we wszystkich powyższych zestawach to:

*F*3, *F*8, *F*17, *F*18, *F*19, *F*21

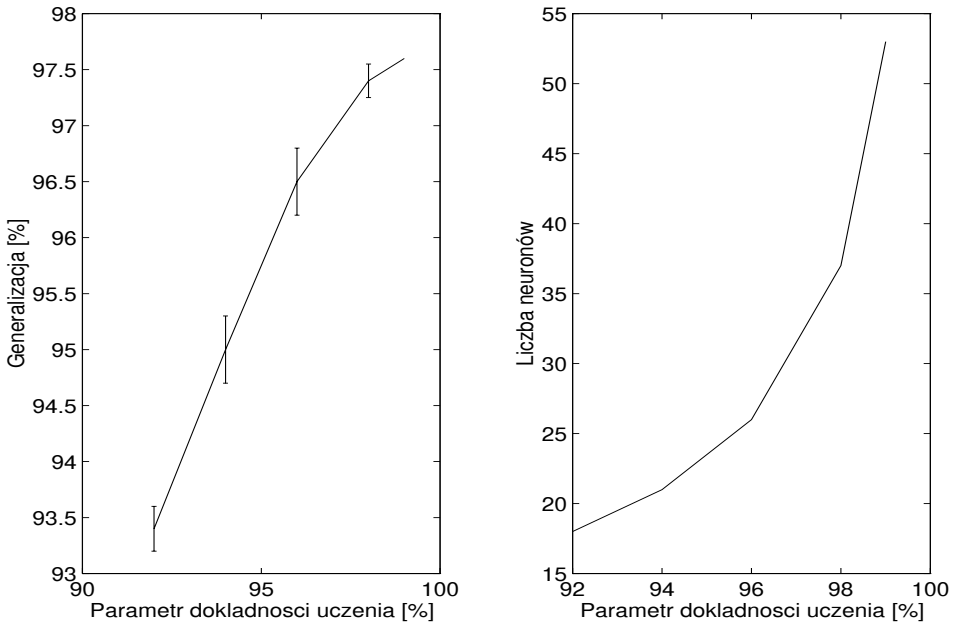
Jest to ten sam zestaw cech, który zastał wykorzystany przez sieć MLP2LN.

Algorytm	Jakość klasyfikacji [%] Na zbiorze testowym	Średnia liczba węzłów

FG	97.4+/-0.3	54
FG+CN=20	98.1+/-0.1	54
FG+R	97.3+/-0.41	62
FG+R+CN=20	97.93+/-0.07	62
FP	99+/-0.15	9
FP+CN=20	99.1+/-0.05	9

Tabela 9-24 Wyniki otrzymane przy użyciu sieci FSM z różnymi funkcjami transferu dla danych tarczycy.

Parametr uczenia sieci FSM został ustalony na podstawie testu 10CV, który wykonano na zbiorze treningowym. Największą generalizację sieci FSM uzyskuje dla parametru uczenia wynoszącego 99%. Podobnie największą generalizację na zbiorze testowym, dla gaussowskiej funkcji aktywacji, uzyskuje się dla parametru uczenia 99%. Porównanie jakości generalizacji na zbiorze testowym w zależności od parametru uczenia sieci FSM przedstawia lewy wykres znajdujący się poniżej. Na prawym wykresie można zaobserwować jak zmienia się liczba węzłów tworzona przez sieć FSM w zależności od parametru uczenia. Podczas wykonywania testu CV jak również przy tworzeniu poniższych wykresów nie użyto parametru uczenia wynoszącego 100%, ponieważ nie udało się nauczyć sieci FSM do takiego poziomu.



Rys 9:5 Lewy rysunek przedstawia zależność dokładności klasyfikacji sieci FSM dla danych tarczycy od parametru dokładności uczenia stosowanego w FSM. Prawy rysunek reprezentuje zależność liczby węzłów od parametru dokładności uczenia.

9.8.3. Dane raka piersi ÑLubljanaî

Dane te zostały pobrane z UCI [6], pochodzą z instytutu onkologii w Lubljanie. W bazie znajduje się 286 przypadków, które opisane są za pomocą 9 cech. Występują dwie klasy zawierające 85 (29.8%) wektorów oraz 201 (70.2%) wektorów.

N. cechy	Opis cechy	Liczba możliwych wartości	Liczba wystąpień wartości brakujących
1	Wiek	9	0
2	Menopauza	3	0
3	Wielkość guza	12	0
4	Liczba przerzutów	13	0
5		2	8
6	Stopień złośliwości	3	0
7	Pierś	2	0
8	Obszar piersi	5	1
9	Naświetlanie	2	0

Tabela 9-25 Spis cech dla danych raka piersi „Lubljana".

Cechy 1,3,4,6 są porządkowe pozostałe nominalne. Dwie cechy zawierają wartości brakujące 6-sta cecha ma 8 wartości brakujących oraz 9-ta cecha ma 1 wartość brakującą. Ekstrakcja reguł za pomocą sieci MLP2LN [18] daje tylko jedną regułę postaci:

$$\text{R1: } F4 > 2 \wedge F6 = 3$$

Reguła powyższa dotyczy klasy przypadków z nawrotami, natomiast druga klasa opisana jest przez jej zaprzeczenie. Jakość klasyfikacji na całym zbiorze wynosi 77%. Reguła R1 jest łatwa do interpretacji: jeżeli liczba guzków jest większa od dwóch i komórki są bardzo złośliwe wówczas wystąpią nawroty. Wydaje się, że jest to najlepszy opis tych danych.

Algorytm	Jakość klasyfikacji [%]	Źródło wyniku
	10CV	
MLP+backprop	73.5+/-9.4	[71]
CART	71.4+/-5.0	[71]
C4.5rules	69.7+/-7.2	[71]
Bayes	75.9	
Weighted networks	68-73.5	
AQ15	66-72	
Default	70.3	

Tabela 9-26 Porównanie wyników zastosowania różnych klasyfikatorów dla danych raka piersi "Lubljana".

Niestety wszystkie klasyfikatory w teście CV osiągają rezultaty bardzo bliskie „Default" i dalekie od wartości osiągniętej przez MLP2LN dla całego zbioru danych. Jest to wynik działania testu CV na niewielkim zbiorze danych. W takich przypadkach wykonanie 10CV może powodować zaburzenie struktury danych. Rezultatem jest osiąganie dużego odchylenia standardowego w kolejnych testach CV, co można zaobserwować w powyższych tabelach. Stąd wynika, że zastosowanie 10CV nie jest dobrym testem dla tych danych. Dużo mniejszy wpływ na zmianę struktury danych ma test LOO (leave one out). Niestety jest to test bardzo kosztowny i nie udało się znaleźć wyników dla tego testu. Aby porównać sieć FSM podobnie jak dla klasyfikatorów w tabeli 9-25 wykonana została 10-krotna kroswalidacja.

Algorytm	Jakość klasyfikacji [%] 10CV	Średnia liczba węzłów
FT	68.02+/-1.82	21
FT+CN=20	68.62+/-0.87	21
FG	70.35+/-1.80	25
FG+CN=20	71.02+/-1.19	25
FG+R	71.62+/-1.8	24
FG+R+CN=20	70.73+/-1.4	24
FP	70.43+/-2.49	9
FP+CN=20	69.8+/-2.26	9

Tabela 9-27 Wyniki otrzymane przy użyciu sieci FSM z różnymi funkcjami transferu dla danych raka piersi „Lubljana".

Pomimo występowania tak prostej reguły żaden z klasyfikatorów nie osiąga rezultatów powyżej „default". MLP+backprop uzyskuje wynik 73.5, jednak przy jednoczesnym bardzo dużym odchyleniu. Rezultaty sieci FSM są porównywalne z pozostałymi klasyfikatorami, mają jednak dużo mniejsze odchylenie standardowe.

9.8.4. Dane raka piersi ÑWisconsinî

Dane raka piersi „Wisconsin" pobrane zostały z UCI [6]. Zawierają 699 przypadków, wśród których występuje 458 przypadków z rakiem złośliwym oraz 245 przypadków z rakiem nie złośliwym. Każdy przypadek opisywany jest przez 9 cech mających wartość w zakresie od 1-10.

N. cechy	Opis cechy	Liczba możliwych wartości	Liczba wystąpień wartości brakującej
1	Rozmiar guzów	10	0
2	Rozmiar komórki	10	0
3	Kształt komórki	10	0
4	Przyleganie	10	0
5		10	0
6		10	16
7		10	0
8		10	0
9		10	0

Tabela 9-28 Spis cech dla danych raka piersi „Wisconsin".

Dla 16 przypadków występuje jedna wartość brakująca. W miejsca wartości brakującej, przy użyciu sieci MLP2LN, wstawiona została wartość średnia dla danej cechy.

Wstępnie za pomocą sieci MLP2LN uzyskano 5 reguł opisujących przypadki z rakiem złośliwym, natomiast przypadki raka nie złośliwego opisane są zaprzeczeniem tych reguł.

R1: $F_2 < 6 \wedge F_4 < 4 \wedge F_7 < 2 \wedge F_8 < 5$ (100)%

R2: $F_2 < 6 \wedge F_5 < 4 \wedge F_7 < 2 \wedge F_8 < 5$ (100)%

R3: $F_2 < 6 \wedge F_4 < 4 \wedge F_5 < 4 \wedge F_7 < 2$ (100)%

R4: $F_2 \in [6,8] \wedge F_4 < 4 \wedge F_5 < 4 \wedge F_7 < 2 \wedge F_8 < 5$ (100)%

R5: $F_2 < 6 \wedge F_4 < 4 \wedge F_5 < 4 \wedge F_7 \in [2,7] \wedge F_8 < 5$ (92.3)%

Pierwsze 4 reguły mają dokładność 100%, ostatnia reguła klasyfikuje tylko 39 przypadków, w tym 3 błędne. Dokładność powyższych reguł na całym zbiorze danych wynosi 96%. Po dokonaniu prostej optymalizacji można otrzymać regułę dla przypadków raka złośliwego postaci

$$F_2 \geq 7 \vee F_7 \geq 6$$

Reguła ta obejmuje 215 przypadków z klasy rak złośliwy oraz 10 przypadków z klasy rak nie złośliwy. W wyniku różnych optymalizacji można otrzymać inny dokładniejszy zestaw reguł. Szczegółowe informacje można znaleźć w pracy [17].

Algorytm	Jakość klasyfikacji [%] 10CV	Źródło wyniku
k-NN	97.14	KMK
IncNet	97.1	[30]
FDA	96.8	[59]
MLP+BP	96.7	[59]
LVQ	96.6	[59]
Naive Bayes	96.4	[59]
LDA	96.0	[59]
Default	65.5	

Tabela 9-29 Porównanie wyników zastosowania różnych klasyfikatorów dla danych raka piersi "Wisconsin".

Najlepszy wynik dla sieci FSM uzyskany został przy zastosowaniu gaussowskiej funkcji aktywacji wraz z obrotami oraz komitetem 30 sieci. Jest to wynik identyczny z wynikiem otrzymanym przez algorytm LVQ. Bardzo zbliżony wynik uzyskuje się również przy zastosowaniu funkcji gaussowskiej z komitetem jak i bez komitetu.

Algorytm	Jakość klasyfikacji [%] 10CV	Średnia liczba węzłów
FT	96.16+/-0.2	13
FT+CN=30	96.35+/-0.32	13
FG	96.42+/-0.33	14
FG+CN=30	96.54+/-0.3	14
FG+R	96.24+/-0.45	14
FG+R+CN=30	96.6+/-0.28	14
FP	95.4+/-0.59	11
FP+CN=30	95.4+/-0.37	11

Tabela 9-30 Wyniki otrzymane przy użyciu sieci FSM z różnymi funkcjami transferu dla danych raka piersi „Wisconsin".

9.8.5. Zapalenie wyrostka robaczkowego

Dane dotyczące zapalenia wyrostka robaczkowego otrzymano od prof. Shalom Weiss z Rutgers University. Zawierają 106 przypadków, opisanych za pomocą 8 cech. Występują dwie klasy mające następujący udział procentowy: 80.2%, 19.8%

W wyniku zastosowania algorytmu MLP2LN otrzymana została następująca reguła:

$$\text{R1: } F3 > 6650 \vee F4 > 12$$

Reguła ta opisuje pierwszą klasę, natomiast druga klasa opisana jest przez jej zaprzeczenie. Jakość poprawnej klasyfikacji przy zastosowaniu tych reguł wynosi 89.6%.

Po zastosowaniu MLP2LN z jednostkami L otrzymana została inna reguła:

$$\text{R1: } F1 > 8400 \vee F4 \geq 4$$

Podobnie jak w poprzednim przypadku druga klasa opisana jest przez zaprzeczenie tej regule. Jest to inny zestaw reguł, który ma dokładnie tą samą jakość klasyfikacji 89.6% na całym zbiorze danych.

Algorytm	Jakość klasyfikacji [%] 10CV	Źródło wyniku
IncNet	90.9	[30]
k-NN	88	KMK
MLP+BP	83.9	KMK
RBF	80.2	KMK
Default	80.2	

Tabela 9-31 Porównanie wyników różnych klasyfikatorów dla danych wyrostka robaczkowego.

Najlepszy wyniki otrzymany przez sieć FSM, przy zastosowaniu gaussowskiej funkcji aktywacji wraz z możliwością obrotu i komitetem sieci, jest prawie taki sam jak wynik *k*-NN. Bardzo dobry wynik uzyskiwany jest również przy użyciu trójkątnej funkcji aktywacji, przy czym liczba węzłów jest o połowę mniejsza.

Algorytm	Jakość klasyfikacji [%] 10CV	Średnia liczba węzłów
FT	86.65+/-1.3	4
FT+CN=20	86.6+/-0.97	4
FG	86.5+/-1.2	5
FG+CN=20	85.95+/-0.84	5
FG+R	86.17+/-1.3	8
FG+R+CN=20	87.62+/-1.2	8
FP	84.5+/-2.5	4
FP+CN=20	80.7+/-1.3	4

Tabela 9-32 Wyniki otrzymane przy użyciu sieci FSM z różnymi funkcjami transferu dla danych wyrostka robaczkowego.

9.8.6. Choroby serca

Dane zostały pobrane z UCI [6]. Występują 303 wektory, z których 164 mają etykietę zdrowy (54.1%), pozostałe oznaczone są etykietą chory. Wszystkie wektory opisane są za pomocą 13 cech.

N. cechy	Opis cechy	Liczba możliwych wartości	Liczba wystąpień wartości brakującej
1	Wiek	Ciągła	0
2	Płeć	2	0
3	Typ bólu w klatce piersiowej	4	0
4	Ciśnienie krwi	Ciągła	0
5	Poziom cholesterolu	Ciągła	0
6		10	0
7	Wynik elektrokardiografii	3	0
8		10	0
9	Czy występują duszności wywołane ćwiczeniami	2	0
10			0
11			0
12	Liczba naczyń głównych zabarwionych przy fluoroskopii	4	5
13	(*thal)*	3	2

Tabela 9-33 Spis cech dla danych reprezentujących choroby serca.

Siedem wektorów zawiera dane brakujące. Dane reprezentują pacjentów: chorych na serce i zdrowych.

Dla tego przypadku reguły otrzymane przy użyciu algorytmu MLP2LN i po zastosowaniu prostej optymalizacji mają następującą postać:

R1: $F13 = 0 \vee F13 = 1 \wedge F12 = 0.0$ (85.5%)
R2: $F13 = 0 \wedge (F12 = 1.0 \vee F12 = 3.0) \wedge F3 \neq 2$ (81.0%)
R3: $(F13 = 0 \vee F12 = 0) \wedge F3 \neq 2$ (85.2%)

Reguły te dają 85.5% poprawności na całym zbiorze danych. Po kolejnych optymalizacjach [17] powstają dwie reguły

R1: $(F13 = 0 \vee F13 = 1) \wedge F12 = 0.0$ (88.5%)
R2: $(F13 = 0 \vee F12 = 0.0) \wedge F3 \neq 2$ (85.2%)

Reguły te wraz z warunkiem *w przeciwnym przypadku druga klasa* dają 89.2% dokładności na całym zbiorze danych.

Algorytm	Jakość klasyfikacji [%]	Źródło wyniku

	10CV	
k-NN	85.1+/-0.5	KMK
FDA	84.2	[59]
Naive Bayes	82.5-83.4	KMK [59]
SNB	83.1	[59]
LVQ	82.9	[59]
Default	54.4	

Tabela 9-34 Porównanie wyników różnych klasyfikatorów dla danych chorób serca.

Algorytm	Jakość klasyfikacji [%] 10CV	Średnia liczba węzłów
FT	80.5+/-0.94	20
FT+CN=20	83.2+/-0.78	20
FG	81.2+/-1.76	24
FG+CN=20	82.98+/-0.6	24
FG+R	82.5 +/- 1.0	24
FG+R+CN=20	81.8+/-0.85	24
FP	74.7+/-1.73	13
FP+CN=20	79.4+/-0.94	13

Tabela 9-35 Wyniki otrzymane przy użyciu sieci FSM z różnymi funkcjami transferu dla danych reprezentujących choroby serca.

Najlepszy rezultat dla sieci FSM uzyskiwany został przy użyciu funkcji trójkątnej wraz z komitetem sieci. Otrzymany rezultat porównywalny jest z wynikiem uzyskiwanym za pomocą SNB czy Naive Bayes. Niestety podobnie jak w poprzednich bazach danych brak jest informacji (za wyjątkiem *k*-NN) dotyczących rozkładu wyników w kilku próbach CV.

10. Zakończenie

Zaprezentowana została sieć MLP2LN, której celem jest ekstrakcja reguł. Cel ten został osiągnięty poprzez modyfikację funkcji błędu stosowanej w sieciach MLP oraz opracowanie konstruktywistycznego algorytmu budowania sieci. Modyfikacja funkcji błędu pozwala uniknąć problemu przeuczenia oraz umożliwia prostą interpretację węzłów ukrytych. Sposób konstrukcji sieci i jej specyficzna architektura rozwiązują problem liczby węzłów ukrytych oraz umożliwiają bardzo szybkie uczenie. Sieć MLP2LN może być zastosowana zarówno do danych dyskretnych, poprzez odpowiednie ich zakodowanie, jak również do danych ciągłych dzięki zastosowaniu jednostek L. Bardzo ważnym elementem występującym w sieci MLP2LN jest również możliwość generowania reguł będących wyjątkami od reguł istniejących. Wszystkie te elementy umożliwiają tworzenie niedużej liczby reguł, oraz reguł z niewielką liczbą przesłanek.

Sieć MLP2LN została zastosowana do analizy danych medycznych. Wyniki przedstawione w rozdziale 9 pokazują, że w wielu przypadkach znajdowane są reguły działające równie dokładnie, co najlepsze klasyfikatory nieregułowe. Co więcej, dla niektórych danych medycznych reguły logiczne odkryte za pomocą sieci MLP2LN są znacznie dokładniejsze niż wyniki otrzymane za pomocą podstawowych odmian sieci MLP (Rprop, QuickProp) oraz wszystkich innych klasyfikatorów (np. dla danych tarczycy). Najprawdopodobniej wynika to z faktu, że lekarze podejmując decyzje kierują się przesłankami, które przybierają właśnie charakter reguł.

Omówiona została modyfikacja sieci propagacji wstecznej błędu umożliwiająca zmianę kształtu obszarów decyzyjnych neuronów poprzez modyfikację ciągu treningowego.

Przedstawiona została sieć FSM. Podobnie jak sieć MLP2LN algorytm sieci FSM jest algorytmem konstruktywistycznym. Poprzez zastosowanie różnych funkcji aktywacji sieć FSM jest zdolna do wytwarzania różnorodnych obszarów decyzyjnych. Umożliwia to lepsze dopasowanie się do danych, a co za tym idzie osiągnięcie lepszej generalizacji. Stosowanie różnych funkcji aktywacji otwiera również różne drogi analizy danych np. interpretację za pomocą reguł logicznych (wystarczy w tym celu zastosować funkcje prostokątne), czy też interpretację opartą o logikę rozmytą (funkcja trójkątna, gaussowska). Zmianę kształtu obszarów decyzyjnych w sieci FSM uzyskuje się również poprzez zastosowanie obrotów w wielowymiarowej przestrzeni. Giętkość tworzenia różnych obszarów decyzyjnych umożliwia zastosowanie sieci FSM w wielu dziedzinach, co potwierdzają przedstawione wyniki. Najlepsze rezultaty uzyskuje się na ogół dla funkcji gaussowskiej jednak są przypadki, gdy najlepszy rezultat osiągany jest dla funkcji prostokątnej 9.8.2, trójkątnej 9.8.6, gaussowskiej z obrotem 9.5. Sieć FSM jest niestabilna, dlatego w celu poprawienia stabilności został zastosowany komitet sieci z głosowaniem. We wszystkich przypadkach zastosowanie komitetu zmniejszyło odchylenie standardowe i w prawie wszystkich przypadkach nastąpiło zwiększenie generalizacji.

11. Spis literatury

[1] R. Andrews, R. Cable, J. Diederich, S. Geva, M. Golea, R. Hayward, Ch. Ho-Stuart, A.B. Tickle, An Evaluation And Comparison Of Techniques For Extracting And Refining Rules From Artificial Neural Networks, QUT NRC technical report, 1995.

[2] M.R. Berthold, K.P.Huber, From radial to rectangular basis functions: a new approach for rule learning from large datasets, University of Karlsruhe, internal report 15-95.

[3] M.J. Berthold and J. Diamond, Boosting the performance of RBF networks with Dynamic Decay Adjustment, W. G. Tesauro, D.S Touretzky and T.K.Leen editors, Advances in Neural Information Processing Systems, 7, Cambridge MA, MIT Press, 1995.

[4] C.M. Bishop, Improving the generalization properties of radial basis function neural networks, Neural Computation 3 (4), 579-588, 1991.

[5] C.M. Bishop, Neural network for pattern recognition, Oxford University Press Inc., New York, 1996.

[6] C.L. Blake and C.J. Merz, UCI repository of machine learning databases. -- http://www.ics.uci.edu/~mlearn/MLRepository.html, 1999.

[7] L. Breiman. Bias-variance, regularization, instability and stabilization. C. M. Bishop, redaktor, Neural Networks and Machine Learning, s. 27–56. Springer-Verlag, 1998.

[8] L. Breiman, Arcing Classifiers, Technical Report 460, Departament of Statistics, University of California, CA, USA, February 1996.

[9] L. Breiman, Bagging predictors, Technical Report 420, Departament of Statistics, University of California, CA, USA, September 1994.

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

[11] M. W. Craven, J.W. Shavlik, Extracting tree-structured representations of trained networks, In: D. Touretzky, M. Mozer, M. Hasselmo, eds, Advances in Neural Information Processing Systems (vol. 8). MIT Press, Cambridge, MA 1996.

[12] Y.L. Cun, J.S. Denker and S.A. Solla, Optimal Brain Damage, Neural Information Processing Systems, Vol. 2, Morgan Kaufman 1990.

[13] A. Czyżewski, Dźwięk cyfrowy. Wybrane zagadnienia teoretyczne, technologia, zastosowania, Akademicka Oficyna wydawnicza EXIT, Warszawa 1998.

[14] B. Denby, The Use of Neural Networks in High energy physics, Neural Computation, Vol. 5, Num. 4, 505-549, July 1993.

[15] W. Duch, R. Adamczak and G.H.F. Diercksen, Distance-based multilayer perceptrons, International Conference on Computational Intelligence for Modelling Control and Automation, 17-19, Vienna, Austria, February 1999.

[16] W. Duch, R. Adamczak, K. Grąbczewski,, A new methodology of extraction, optimization and application of crisp and fuzzy logical rules, IEEE Transactions on neural networks, w druku.

[17] W. Duch, R. Adamczak, K. Grąbczewski and G. Żal, A hybrid method for extraction of logical rules from data, Colloquia on Artificial Inteligence, Łódź, Poland, pp. 61-82, 28-30.09.1998.

[18] W. Duch, R. Adamczak and K. Grąbczewski, Methodology of extraction, optimization and application of logical rules, Intelligent Information Systems VIII, Ustroń, Poland, pp. 22-31, 14-18.06.1998.

[19] W. Duch, G. H. F. Diercksen, Feature Space Mapping as a universal adaptive system, Computer Physics Communications, Vol 87, pp. 341-371, 1995.

[20] W. Duch, N. Jankowski, Survey of neural transfer functions, Neural Computing Surveys, Vol. 2, pp. 163-213, 1999.

[21] B. Efron, R. Tibshirani, Cross-Validation and the Bootstrap: Estimating the Error Rate of a Prediction Rule, Technical Report, May 1995.

[22] S.E. Fahlman, C. Lebier, The Cascade-Corelation Learning Architecture, Technical Report CMU-CS-90-100, School of Computer Science, Carnegie Mellon University, August 1991.

[23] S.E. Fahlman, Faster-learning variations on back-propagation: An empirical study. In T. J. Sejnowski G. E. Hinton and D. S. Touretzky, editors, Connectionist Models Summer School, San Mateo, CA, Morgan Kaufmann. 1988.

[24] E. Gatnar, Symboliczne metody klasyfikacji, Wydawnictwo Naukowe PWN S.A., Warszawa 1998.

[25] P. Greczy, S. Usui, Rule extraction from trained neural networks. Int. Conf. on Neural Information Processing, New Zeland, Vol. 2, pp. 835-838, Nov. 1997.

[26] B. Hassibi, D. G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. Advances in Neural Information Processing Systems 5. Morgan Kaufmann, 1993.

[27] B. Hassibi, D.G. Stork, G. Wolff, Optimal Brain Surgeon: Extensions performance comparision, ICNN, San Francisco 1993.

[28] S. Haykin, Neural networks a comprehensive foundation, Macmillan Publishing Company, 1994.

[29] Hertz, Wstęp do obliczeń neuronowych, WNT Warszawa, 1993.

[30] M. Ishikawa, Structural learning with forgetting, Neural Networks 9, 509-521, 1996.

[31] N. Jankowski. Ontogenic neural networks and their applications to classification of medical data, PhD thesis, Department of Computer Methods, Nicholas Copernicus University, Toruń, Poland, 1999.

[32] R.D. King, A. Srinivasan, M.J.E. Sternberg, Relating chemical activity to structure: an examination of ILP successes, New Gen. Comput., 13(3,4):411-433, 1995.

[33] G. King, J. Honaker, A. Joseph, K. Scheve, Analyzing incomplete political science data: an alternative algorithm for multiple imputation, 1999.

[34] T. Kohonen, H. Hynninen, J. Kangas, H. Laaksonen, and K. Torkkola. LVQ-PAK: The learning vector quantization program package. Technical Report A30, Helsinki University of Technology, Laboratory of Computer and Information Science, FIN-02150 Espoo, Finland, 1996.

[35] J.B. Kruskal, Multidimensional scaling by optimizing goodness of-fit to a non-metric hypothesis, Psychometrika, 29, str. 1-27, 1964.

[36] O. Lahav , A. Naim , L. Jr. Sodre and M.C. Storne-Lombardi, Neural Comptutation as a tool for galaxy classification: methods and examples. -- Institute od Astronomy, Cambridge, Technical report CB3 OHA, 1995.

[37] L.E. Lehman, Theory of point estimation, Wiley, New York, 1983.

[38] D. J. C. MacKay, Probable networks and plausible predictions --- a review of practical Bayesian methods for supervised neural networks. Network: Computation in Neural Systems 6: 469—505, 1995.

[39] J.B. MacQueen, Some Methods for Classification and Analysis of Multivariate Observations, Proceedings of the Fifth Berkeley Symposiuon Mathematical Statistics and Probability, 1, 281-297, 1967

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

[41] J. Moody, C. J. Darken, Fast learning in networks of locally-tuned processing units, Neural Computation 1, 281-294, 1989.

[42] M. Mulholland, D.B. Hibbert, P.R. Haddad, P. Parslov, A comparision of classification in artificial intelligence, induction versus a self-organising neural networks, Chemometric and intelligent laboratory systems 30, pp. 117-128, 1995.

[43] M. Mullholand, D.B. Hibbert, P.R. Haddad, C. Sammut, Application of the C4.5 classifier to building an expert system for ion chromatography, Chemometrics and Intelligent Laboratory Systems 27, pp. 95-104, 1995.

[44] M. Opper and O. Winther, Mean Field methods for classification with Gaussian processes, Advances in Neural Information Processing Systems 11 (NIPS'98), eds. M. S. Kearns, S. A. Solla, and D. A. Cohn, MIT Press, Cambridge, MA (1999).

[45] S. Osowski, Sieci neuronowe w ujęciu algorytmicznym, Wydawnictwo Naukowo-Techniczne, Warszawa 1996.

[46] M.P. Perrone, L. N. Cooper, When networks disegree: ensemble methods for hybrid neural networks, R. J. Mammone (Ed.), Artificial Neural Networks for Speech and Vision, pp. 126-142 London: Chapman & Hall.

[47] J. Platt, A resource-allocating network for function interpolation, Neural Computation 3, 213-225, 1991.

[48] T. Poggio and F. Girosi, A theory of networks for aproximation and learning, Raport instytutowy A. I. Memo 1140, MIT, Massachusetts, 1989.

[49] J.R. Quinlan, Improved use of Continuous attributes in C4.5, Journal of Artificial Intelligence Research 4, pp. 77-90, 1996.

[50] M. D. Richard, R.P. Lippmann, Neural networks classifiers estimate bayesian a posteriori popbabilities, Neural computation 3, 461-468

[51] Z. Ramadan, M. Mulholand, D.B. Hibbert, P. Preston, P. Compton, P.R. Haddad, Towards an expert systems in ion-exclusion chromatography by means of multiple classification ripple-down rules, Journal of Chromatography A, 804, pp. 29-35, 1998.

[52] J.S. Rao, R. Tibshirani, The out-of-bootstrap method for model averaging and selection, Technical Report, May 1997.

[53] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, Parallel Distributed Processing: Explorations in the microstructure of cognition; Vol. 1: Foundations, Cambridge, Massachusetts, The MIT Press. 1986.

[54] J. L. Schafer, M. K. Olsen, Multiple imputation for multivariate missing-data problems, a data analyst's perspective, 1998.

[55] R.J. Schalkof, Pattern Recognition: statistical, structural and neural approaches, R.R. Donnelley & Sons, 1992.

[56] A. Srinivasan, S.H. Muggleton, R.D. King, M.J.E. Sternberg, The predictive toxicology evaluation challenge, In Proceedings of Fifteen International Conference on Artificial Intelligence (IJCAI-97). Morgan Kaufman, Los Angeles, CA, 1997.

[57] R. Setiono, H. Liu, Understanding neural networks via rule extraction, ln: Proc.of the 14th Int. Joint Conference on Artificial Intelligence, Montreal, Quebeck. Morgan Kaufmann, pp. 480-485, 1995.

[58] W. Sobczak, W. Malina, Metody selekcji informacji, WNT 1985.

[59] B. Ster and A. Dobnikar, Neural networks in medical diagnosis: Comparison with other methods. In A. Bulsari et al., editor, Proceedings of the International Conference EANN '96, pp. 427-430, 1996.

[60] T.Thiemann and V.Vill, Preliminary communication development of an incremental systems for the prediction of the nematic-isotropic phase transition temperature of liquid crystals with two aromatic rings, Liquid crystals, Vol. 22, No.4, 519-523, 1997.

[61] G. Thimm, E. Fiesler, High Order and Multilayer Perceptron Initialization, IEEE Transaction on Neural Networks, 1996.

[62] S. Thrun, Extracting rules from artificial neural networks width distributed representations, ln: G. Tesauro, D. Touretzky, T. Leen, eds, Advance in Neural Infomtation Processing Systems 7, MIT Press, Cambridge, MA, 1995.

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

[64] M.E. Tipping, Topographic mappings and fead-forward neural networks, PhD thesis, University of Aston in Birmingham, 1996.

[65] G. Towell, J. Shavlik, Extracting refined rules from knowledge-based neural networks, Machine Learning 13, 71-101, 1993.

[66] K. Tumer and J. Ghosh, Linear and Order Statistics Combiners for Pattern Classification, in Combining Artificial Neural Nets, A. Sharkey (Ed.), Springer-Verlag, pp. 127-162, 1999.

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

[68] D.R. Wilson, T.R. Martinez, Improved Heterogeneous Distance Functions, Journal of Artificial Intelligence Research 6, pp. 1-34, 1997.

[69] D. H. Wolpert, Stacked generalization, Neural networks, 5:241-245, 1992.

[70] S. De Backer, A. Naud, P. Scheuders, Non-linear dimensionality reduction techniques for unsupervised feature extraction, Pattern Recognition Letters 19, str. 711-720, 1998.

[71] F. Zarndt, A Comprehensive Case Study: An Examination of Machine Learning and Connectionist Algorithms, MSc Thesis, Dept. of Computer Science, Brigham Young University, 1995.

[72] J. Żurada, M. Barski, W. Jędruch, Sztuczne sieci neuronowe, Wydawnictwo naukowe PWN, Warszawa 1996.