AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA
WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI, INFORMATYKI I ELEKTRONIKI
KATEDRA AUTOMATYKI

ROZPRAWA DOKTORSKA

MODELOWANIE I SYMULACJA WYBRANYCH PARAMETRÓW MEDYCZNYCH DLA PRZYPADKÓW Z NIEWYDOLNOŚCIĄ ODDECHOWĄ

MGR INŻ. PIOTR WAIS

Promotor: Prof. dr hab. inż. Wiesław Wajs

Kraków, 2008

Streszczenie

Tematem rozprawy doktorskiej jest modelowanie i symulacja wybranych parametrów medycznych dla przypadków z niewydolnością oddechową. Zagadnienie to jest ważnym problemem dla pacjentów przyjętych do Kliniki do drugiej doby życia z powodu niewydolności układu oddechowego. Do modelowania i symulacji stanów chorobowych wykorzystano metody komputerowe z dziedziny sztucznej inteligencji. Przeprowadzone w ramach pracy badania dotyczyły w przeważającym stopniu określenia przydatności metod adresowanych do dwóch różnych problemów.

Pierwszy problem związany jest z modelowaniem przewidywanych stanów chorobowych pacjenta. Stany te są generowane za pomocą metody opartej na algorytmie Artificial Immune System wyprowadzonym z obserwacji systemu immunologicznego kręgowców. Dane potrzebne do wygenerowania stanów chorobowych w systemie komputerowym pochodzą z oryginalnie zaprojektowanej bazy danych. W pracy zaproponowano i rozwinięto metodę nazwaną modelem postępowania lekarza. Metoda ta służy jako metoda referencyjna dla metody opartej na algorytmie Artificial Immune System. Metoda nazwana „model postępowania lekarza" zapewnia możliwość symulowania zdarzeń przyszłości, jako narzędzie służące do predykcji stanów chorobowych. Następuje to poprzez badanie wpływu określonych decyzji lekarza na przebieg ważnych parametrów medycznych w przewidywanych stanach chorobowych. Obok znanego algorytmu De Castro zaprojektowano oryginalny algorytm służący do porównania rezultatów obliczeń.

Innym poruszonym w pracy problemem jest metoda klasyfikacji zdarzeń. Metoda ta znana w literaturze jako Support Vector Machine jest używana do klasyfikacji parametrów medycznych, które są pomocne w zagadnieniu określenia stanu przeżycia lub zgonu pacjenta z niewydolnością oddechową. Metoda Support Vector Machine jest porównywana pod względem skuteczności i dokładności z metodą Sztucznych Sieci Neuronowych zaproponowanych przez Specht'a. Metoda Sztucznych Sieci Neuronowych jest w pracy traktowana jako metoda referencyjna dla metody Support Vector Machine.

W pracy przeprowadzono liczne badania na oryginalnym materiale zgromadzonym w bazie danych. Wyniki dokumentowano w formie wykresów i tabel dla wybranych istotnych parametrów medycznych. W sposób obszerny porównano dostępne i używane w literaturze metody pod względem dokładności uzyskanych wyników dla przyjętych definicji błędów w porównaniu do rezultatów zgromadzonych w bazie danych, które przyjęto jako wartości dokładne. W pracy wykorzystano trzy znane metody obliczeniowe: SVM, SSN i AIS oraz zaproponowano

dwie oryginalne metody: AIS algorytm 2 oraz MPL. Zastosowanie tych metod umożliwiło porównanie uzyskanych wyników. Praca w swojej warstwie poznawczej dokumentuje przydatność metod obliczeniowych z obszaru Sztucznej Inteligencji do modelowania i symulacji przypadków medycznych dotyczących niewydolności oddechowej dzieci przyjętych do Kliniki w pierwszej lub najpóźniej drugiej dobie życia.

Spis skrótów i oznaczeń

AIS - Artificial Immune System
SVM - Support Vector Machines
MPL - Model Postępowania Lekarza
SSN - Sztuczna Sieć Neuronowa *(ang. Artificial Neural Network)*
NIS - Neonatal Information System
RDS - Zespół zaburzeń oddychania *(ang. Respiratory Distress Syndrome)*
pO_2 - ciśnienie parcjalne tlenu we krwi
pCO_2 - ciśnienie parcjalne dwutlenku węgla we krwi
pH - odczyn kwasowo-zasadowy krwi
HCO_3- stężenie wodorowęglanów we krwi
BE -nadmiar zasad *(ang. base excesse)*
FiO_2 - parametr nastawy respiratora, jest to odsetek tlenu w mieszaninie oddechowej
pO_2/FiO_2 - wskaźnik tlenowy
$CPAP$ - ciągle dodatnie ciśnienie w drogach oddechowych
(ang. Continuous Pressure Air Papline)
IMV - przerywana wentylacja wymuszona
(ang. Intermittent Mandatory Ventilation)
$SIMV$ - synchroniczna, przerywana wentylacja wymuszona
(ang. Synchronized Intermittent Mandatory Ventilation)
HFO - wentylacja wysokimi częstotliwościami
(ang. High Frequency Oscillation)
RR - częstość oddechów *(ang. Respiratory Rate)*
TV - objętość oddechowa *(ang. Tidal Valume)*
PIP - ciśnienie szczytowe wdechu *(ang. Peak Inspiratory Pressure)*
MAP - średnie ciśnienie w drogach oddechowych *(ang. Mean Airway Pressure)*
IVH - wylew śródczaszkowy
NEC - martwicze zapalenie jelit
PDA - obecność przetrwałego przewodu tętniczego
(ang. Patent Ductus Arteriosis)
BPD - dysplazja oskrzelowo-płucna *(ang. Bronchopulmonary Dysplasia)*
SURF - zastosowanie surfaktantu
RBF - Radial Basis Function

Spis treści

Rozdział 1

Wstęp

Wcześniactwo, obok zamartwicy i wad wrodzonych, stanowi główny problem medycyny wieku noworodkowego. Ocenia się, że w krajach Europy zachodniej około 4-8% wszystkich ciąż rozwiązywanych jest przedwcześnie. Sam poród przedwczesny pociąga za sobą zwiększone ryzyko wystąpienia różnych zaburzeń u noworodka wiążących się z niepełnym przygotowaniem dziecka do życia pozamacicznego. Niedojrzałość narządów i układów organizmu ma charakter anatomiczny i czynnościowy. Na stan dziecka po urodzeniu wpływają także czynniki, które wyzwoliły poród przedwczesny, np. zakażenia wewnątrzmaciczne. Szybki postęp neonatologii zaowocował zwiększoną przeżywalnością noworodków urodzonych przedwcześnie. Dzięki znajomości patofizjologii zaburzeń związanych z wcześniactwem, dostępności nowoczesnych, nieinwazyjnych technik monitorowania, mikrometod w badaniach analitycznych oraz udoskonaleniu technik sztucznej wentylacji, możliwe jest przeżycie dzieci z masą urodzeniową 500g urodzonych w 22-24 tygodniu ciąży. Kluczowe znaczenie dla przeżycia dziecka urodzonego przedwcześnie ma układ oddechowy. Zespół zaburzeń oddychania (Respiratory Distress Syndrome - RDS) jest powikłaniem wcześniactwa wynikającym z niedojrzałości anatomicznej i czynnościowej płuc, związanej głównie z niedoborem surfaktantu - substancji zmniejszającej napięcie powierzchniowe pęcherzyków płucnych. Objawy RDS występują w ciągu pierwszych godzin życia. W niektórych przypadkach mają charakter łagodny, wymagają jedynie leczenia objawowego i ustępują samoistnie. Często RDS przebiega z nasiloną niewydolnością oddechową, która mimo stosowania zaawansowanych technik wspomagania oddychania i podawania surfaktantu nasila się prowadząc do zgonu dziecka. Opieka nad wcześniakiem z zaburzeniami oddychania jest niezwykle złożona. Polega na ciągłym nadzorze funkcji praktycznie wszystkich układów i narządów. W ocenie stanu dziecka brane są pod uwagę parametry statyczne, takie jak: urodzeniowa masa ciała, wiek płodowy, punktacja w skali Apgar. W ocenie stanu mogą być także brane pod uwagę parametry dynamiczne mierzone cyfrowo lub analogowo. Są to wartości gazometrii krwi tętniczej, morfologii krwi, parametry biochemiczne, wyniki badań obrazowych oraz parametry dynamiczne mierzone cyfrowo w niewielkich odstępach czasu albo analogowo w sposób ciągły, następnie konwertowane do postaci cyfrowej. Do parametrów mierzonych dynamicznie można zaliczyć wysycenie hemoglobiny tlenem, zapis czynności serca, ciśnie-

nie tętnicze, parametry mechaniki oddychania. Pełna ocena stanu noworodka uwzględnia także stosowane metody leczenia: parametry wentylacji mechanicznej, stosowanie leków (surfaktant, aminy katecholowe) i wykonywanie zabiegów (np. podwiązanie przewodu tętniczego). Ocena stanu aktualnego wraz z przewidywaniem dalszego przebiegu niewydolności oddechowej stanowi podstawę do podejmowania przez lekarza decyzji terapeutycznych. Pierwsze dwa tygodnie życia noworodków urodzonych z bardzo niską masą urodzeniową, czyli mniejszą niż < 1500[g] cechuje bardzo duża dynamika zaburzeń. Właściwą miarą czasu w tym okresie życia są godziny, a nawet minuty. Jednoczesna, ciągła analiza wielu zmieniających się parametrów wymaga ogromnego doświadczenia i precyzji. Ocena lekarska, mimo że w warunkach intensywnej terapii w większości przypadków sprowadza się do analizy parametrów mierzalnych, jest subiektywna. Mają na nią wpływ takie czynniki jak: najnowsze doniesienia literaturowe i własne doświadczenie lekarza. Metody matematyczne, dzięki zastosowaniu technik komputerowych ułatwiają przeprowadzanie analizy i pozwalają na połączenie wiedzy teoretycznej (np. algorytmów decyzyjnych) i empirycznej (zapis przebiegu choroby wielu uprzednio hospitalizowanych pacjentów w komputerowej bazie danych). Zastosowanie właściwych narzędzi matematycznych umożliwia opracowanie oprogramowania analizującego parametry statyczne i dynamiczne, pozwalającego na modelowanie przebiegu złożonych zjawisk biologicznych. Tego typu programy komputerowe mogą przedstawiać aktualny stan pacjenta na tle innych, uprzednio hospitalizowanych noworodków, przewidywać dalszy przebieg zaburzeń oraz umożliwiać symulację zastosowania konkretnych metod leczenia wraz z prognozą ich efektu. Narzędzia posiadające te właściwości stanowiłyby istotną pomoc dla lekarza w procesie podejmowania decyzji.

1.1 Informatyczne systemy modelowania i klasyfikacji - AIS, SVM

W ostatnich kilku latach nastąpił wzrost zainteresowania biologią jako źródłem inspiracji do rozwiązywania wielu problemów obliczeniowych z zakresu przetwarzania danych, inteligencji obliczeniowej (ang. Computational Intelligence) i optymalizacji. Motywacją tego zainteresowania jest dokonanie wyodrębnienia mechanizmów, które są wykorzystywane przez systemy naturalne i próba ich adaptacji do efektywnego rozwiązywania problemów z wyżej wymienionych dziedzin. Przykładem efektywnej adaptacji naturalnych mechanizmów mogą być sztuczne sieci neuronowe, algorytmy genetyczne, a ogólnie algorytmy ewolucyjne, które obecnie znajdują szerokie zastosowanie w różnych dziedzinach, a zwłaszcza modelowaniu złożonych zjawisk występujących w praktyce medycznej. W latach osiemdziesiątych i dziewięćdziesiątych poprzedniego stulecia zainteresowano się systemem immunologicznym kręgowców, a szczególnie człowieka. Wiele z mechanizmów, które funkcjonują w naturalnym systemie immunologicznym jest z powodzeniem wykorzystywane do rozwiązywania wielu problemów z zakresu klasyfikacji danych wielowymiarowych, predykcji szeregów czasowych, optymalizacji funkcji, ekstrakcji wiedzy. Budowę i funkcjonowanie systemu immunologicznego człowieka

wraz z jego interakcjami porównuje się ze złożonością układu nerwowego, a w szczególności mózgu. Dzięki rozwojowi biologii molekularnej i genetyki zostały częściowo poznane i wyjaśnione mechanizmy funkcjonowania systemu odpornościowego. Okazało się, że opiera się ono na kilku podstawowych mechanizmach, które są nie tylko interesujące pod względem biologicznym, ale także mogą z powodzeniem zostać zaadaptowane w informatyce jako nowe techniki obliczeniowe. *Sztuczny system immunologiczny* (AIS) może zostać zdefiniowany jako struktura lub mechanizm funkcjonowania bazujący na systemie odpornościowym kręgowców. Wszystkie kręgowce są wyposażone w mechanizmy, które umożliwiają im obronę przed różnego typu chorobami. Mechanizmy te są jednym z rezultatów procesu ewolucji i jak się obecnie okazało w najbardziej zaawansowane systemy immunologiczne są wyposażone kręgowce[18]. System immunologiczny kręgowców jest szczególnie interesujący z powodu jego szczególnych własności obliczeniowych [3, 2, 32].

Innym przydatnym modelem obliczeniowym stosowanym w pracy jest metoda znana w literaturze jako *Support Vector Machines* (SVM) zaproponowana przez Vapnika [23, 24]. SVM może być używany do klasyfikacji i regresji. Centralnym elementem metody SVM jest konstrukcja optymalnej hiperpłaszczyzny, której zadaniem jest odseparowanie danych należących do różnych klas, z możliwie największym marginesem rozdzielającym klasyfikowane zbiory. Przez margines rozdzielający rozumie się tu odległość pomiędzy optymalnie poprowadzoną hiperpłaszczyzną i najbliższym punktem przestrzeni po obu stronach hiperpłaszczyzny rozdzielającej klasyfikowane zbiory.

1.2 Cel pracy

Celem pracy jest zastosowanie matematycznych metod sztucznej inteligencji w modelowaniu przebiegu niewydolności oddechowej u noworodków urodzonych przedwcześnie z zespołem zaburzeń oddychania.

Cele szczegółowe:

- predykcja przebiegu wskaźnika pO_2/FiO_2 odzwierciedlającego nasilenie niewydolności oddechowej z wykorzystaniem *Sztucznej Sieci Immunologicznej (AIS)*
- ocena ryzyka wczesnego zgonu pacjenta do końca 2 tygodnia życia noworodka z zaburzeniami oddychania z wykorzystaniem metody *Support Vector Machines (SVM).*

Pozytywna ocena predykcji wskaźnika i ryzyka zgonu może stanowić podstawę do budowy oprogramowania wspomagającego lekarza w procesie podejmowania decyzji odnośnie metod leczenia pacjenta.

1.3 Teza pracy

Sztuczna Sieć Immunologiczna (AIS) jest narzędziem przydatnym do predykcji przebiegu wskaźnika pO_2/FiO_2 odzwierciedlającego nasilenie niewydolności od-

dechowej, a metoda Support Vector Machines (SVM) jest metodą przydatną do oceny ryzyka wczesnego zgonu noworodka, u którego występują zaburzenia oddychania.

1.4 Struktura pracy

Część pierwsza pracy przedstawia opis materiału badawczego: źródło danych, charakterystykę badanych pacjentów oraz opis i charakterystykę parametrów przypadków chorobowych.

Druga część opisuje metody badawcze wykorzystywane do predykcji wskaźnika pO_2/FiO_2 i oceny ryzyka zgonu. Zaproponowano metodę Sztucznych Systemów Immunologicznych oraz metodę Support Vector Machines.

Część trzecia pracy opisuje wyniki przeprowadzonych badań i porównanie osiągniętych rezultatów.

W części czwartej zamieszczono podsumowanie oraz wnioski końcowe.

1.5 Materiał i źródła danych

Dane wykorzystane w pracy pochodzą z bazy danych Neonatal Information System (NIS), w której gromadzone są informacje o hospitalizowanych dzieciach. Baza ta znajduje się na Oddziale Intensywnej Terapii Noworodka Polsko-Amerykańskiego Instytutu Pediatrii Collegium Medicum Uniwersytetu Jagiellońskiego. Dane były gromadzone w latach 1991-2007. Baza danych NIS była wykorzystywana jako źródło danych medycznych w wielu pracach [11, 26, 19, 10, 12, 25, 27, 29, 30, 31].

Przeprowadzane badania fizykalne, laboratoryjne i rejestracja wskazań aparatury medycznej dostarczają informacji o parametrach opisujących aktualny stan zdrowia pacjenta. Dokonywane są one kilka razy dziennie z jednoczesnym zapisem informacji w dokumentacji historii choroby pacjenta i rejestracją danych w szpitalnej bazie danych. W oparciu o uzyskaną informację podejmowana jest decyzja o stosowanych środkach leczenia. Decyzje mogą dotyczyć doboru nastaw respiratora oraz decyzji o podaniu leku i terminu jego podania. Dane o tych faktach rejestrowane są w bazie danych. Działania medyczne następują kolejno po sobie, dostarczając do bazy danych informacji o historii choroby określonego pacjenta. Dla potrzeb niniejszej pracy pobierane są dane dotyczące pacjentów z zaburzeniami oddychania w niewielkich odstępach czasu. Są to dane o charakterze statycznym takie jak: urodzeniowa masa ciała, wiek płodowy. Szczególne znaczenie mają dane o charakterze dynamicznym: gazometria krwi, terminy podania leków, zmiana nastawy respiratora. Informacje o charakterze dynamicznym pochodzą z nowej bazy danych o nazwie New Neonatal Information System.

Rozdział 2

Charakterystyka badanych pacjentów

Z bazy danych *Neonatal Information System (NIS)* wybrano pacjentów urodzonych przedwcześnie z masą ciała mniejszą lub równą 1500[g] i przyjętych do kliniki do końca drugiej doby życia. W zależności od nasilenia zaburzeń oddychania pacjentów podzielono na cztery grupy biorąc pod uwagę wartość wskaźnika pO_2/FiO_2 w chwili przyjęcia dziecka do kliniki.

Grupa pierwsza obejmuje dzieci zdrowe z niewielkimi zaburzeniami oddechowymi. Grupa ta jest opisana relacją

$$pO_2/FiO_2 > 300[mmHg]$$

i charakteryzuje się brakiem wsparcia oddychania.

Grupa druga dzieci z umiarkowanymi zaburzeniami oddechowymi. Grupa ta jest opisana relacją

$$pO_2/FiO_2 = od\ 200\ do\ 300[mmHg]$$

i obejmuje przypadki wymagające wsparcia oddechowego w postaci: wąsów tlenowych, CPAP, IMV/SIMV.

Grupa trzecia obejmuje dzieci z ciężkimi zaburzeniami oddechowymi. Grupa ta jest opisana relacją

$$pO_2/FiO_2 = od\ 100\ do\ 199[mmHg]$$

oraz wsparciem oddechowym – HFO, IMV/SIMV.

Grupa czwarta to dzieci ze skrajnymi zaburzeniami oddechowymi. Grupa ta charakteryzuje się wysokim ryzykiem zgonu z przyczyn oddechowych i jest opisana relacją

$$pO_2/FiO_2 < 100[mmHg]$$

oraz wsparciem oddechowym - IMV/SIMV, HFO.

2.1 Charakterystyka parametrów

Do badań wybrane zostały parametry gazometrii krwi tętniczej: pO_2, pH, pCO_2, HCO_3, BE. Są to wielkości opisujące gospodarkę kwasowo-zasadową i wymianę gazową w płucach.

pO_2 i pCO_2 to odpowiednio ciśnienie parcjalne tlenu i ciśnienie parcjalne dwutlenku węgla, pH oznacza odczyn kwasowo-zasadowy krwi, HCO_3 i BE to parametry opisujące komponentę metaboliczną istotną w utrzymaniu właściwego pH krwi. Pojedynczym, najlepiej całościowo opisującym parametrem wskazującym ogólny stan organizmu pacjenta jest pH. Nieskompensowane zaburzenia wymiany gazowej oraz zaburzenia w metabolizmie powodują obniżenie pH. Poniżej przedstawiono krótką charakterystykę parametrów [10].

pH odczyn krwi jest parametrem zależnym od stanu pracy układu oddechowego, ale także od czynności układu krążenia, nerek i metabolizmu ustroju. Jest to wskaźnik ogólny, który można traktować jako wskaźnik ogólnego stanu pacjenta. Wartości prawidłowe pH wahają się od 7.35 do 7.45.

pCO_2 ciśnienie parcjalne dwutlenku węgla ściśle koreluje ze stopniem wydolności układu oddechowego. Ciśnienie parcjalne dwutlenku węgla zależne jest przede wszystkim od wentylacji minutowej. Wzrost tej wartości powoduje spadek pCO_2. Wzrost ciśnienia parcjalnego dwutlenku węgla jest związany z zaburzeniami oddychania powiązanymi z obniżoną wentylacją płuc. W ciężkich zaburzeniach oddychania związanymi z upośledzeniem dyfuzji gazów przez barierę pęcherzykowo-włośniczkową dochodzi także do upośledzenia wymiany dwutlenku węgla. Wzrost wartości pCO_2 obniża pH krwi. U pacjentów wentylowanych mechanicznie wartość pCO_2 zależy głównie od częstości oddechów RR *(ang. Respiratory Rate)* i objętości oddechowej TV. Wartości prawidłowe pCO_2 wahają się od 35 do 45 [mmHg].

pO_2 ciśnienie parcjalne tlenu ściśle koreluje ze stopniem wydolności układu oddechowego. Prawidłowa wymiana tlenu w płucach zależy głównie od odpowiedniego stosunku wentylacji do perfuzji oraz od przepuszczalności bariery pęcherzykowo-włośniczkowej. Zaburzenie stosunku wentylacji do perfuzji obserwujemy głównie w przypadku niedodmy. Pogrubienie bariery pęcherzykowo-włośniczkowej w pierwszym rzędzie upośledza wymianę tlenu, dopiero dalsze ograniczenie jej przepuszczalności doprowadza do upośledzenia wymiany CO_2. U pacjentów, u których stosujemy wspomaganie oddychania, pO_2 zależy od stężenia tlenu w mieszaninie oddechowej FiO_2 oraz od stopnia rozprężenia płuc. Stopień rozprężenia płuc u pacjentów wentylowanych mechanicznie uwarunkowany jest odpowiednim ciśnieniem w drogach oddechowych PIP - ciśnienie szczytowe wdechu *(Peak Inspiratory Pressure)* oraz od MAP *(Mean Airway Pressure)* opisujący średnie ciśnienie w drogach oddechowych. Niedobór tlenu w organizmie powoduje nasilenie metabolizmu beztlenowego, a więc także zwiększonej produkcji kwasu mlekowego i spadku pH krwi. U dzieci urodzonych przedwcześnie wartości prawidłowe pO_2 wynoszą od 40 do 50[mmHg].

HCO_3 stężenie wodorowęglanów we krwi zależy od metabolizmu ustroju oraz od funkcji nerek. W przypadku dysproporcji między wydolnością układu dostarczania tlenu do tkanek, a potrzebami metabolicznymi ustroju pacjenta dochodzi

do spadku stężenia dwuwęglanów i kwasicy metabolicznej. Wielkość ta, pośrednio zależy od pO_2. Stężenie dwuwęglanów we krwi wyraża się w [mmol/l]. Wartości prawidłowe tej wielkości mieszczą się w przedziale 21-25 [mmol/l].

Parametr nastawy respiratora oznacza się jako FiO_2. Wartości gazometrii krwi tętniczej zawsze należy rozpatrywać w kontekście stosowanego wsparcia oddechowego. Prawidłowe wartości u pacjenta wspartego mechaniką respiratora i oddychającego w 100% tlenem mają inne znaczenie niż te same wielkości odnoszące się do pacjenta oddychającego samodzielnie mieszaniną gazową o 21% zawartości tlenu. FiO_2 jest to odsetek tlenu w mieszaninie oddechowej (*Fraction of inspired Oxygen*). Parametr ten bezpośrednio wpływa na wartość pO_2 w krwi tętniczej, waha się od 0.21 do 1.0.

Wskaźnik tlenowy pO_2/FiO_2 [mmHg] łączy dwie wielkości. Parametr nastawy respiratora FiO_2 oraz pozostający w bezpośredniej zależności od niego parametr gazometrii krwi pO_2. Parametr ten opisuje nasilenie zaburzeń oddechowych. Cechą tego parametru jest prostota i możliwość stosowania zarówno w odniesieniu do pacjentów oddychających samodzielnie jak i korzystających ze wsparcia respiratora. Wadą jest brak ujęcia innych wielkości charakteryzujących oddychanie wspomagane, na przykład: średniego ciśnienia w drogach oddechowych MAP.

Parametr masa urodzeniowa ciała *(ang. birthweight)* jest użyteczny w modelowaniu zaburzeń oddychania. Stopień nasilenia problemów oddechowych w większości przypadków zależy od stopnia niedojrzałości dziecka. Urodzeniowa masa ciała jest mierzona u każdego noworodka i w większości przypadków dość dobrze odzwierciedla stopień dojrzałości noworodka.

Parametr płeć dziecka. W piśmiennictwie istnieje wiele doniesień dotyczących wpływu różnych czynników na nasilenie zaburzeń oddychania u wcześniaków. Jednym z takich czynników jest płeć męska, która kojarzy się z cięższym przebiegiem RDS.

Parametr wiek płodowy *(ang. gestational age)* bezpośrednio oddaje stopień niedojrzałości wcześniaka, jednak nie zawsze ten parametr jest możliwy do jednoznacznego określenia.

Parametr zastosowanie surfaktantu *(medications - surfactant)* ma znaczenie w modelowaniu zjawisk występujących podczas zaburzeń oddychania. Surfaktant jest lekiem, który stosowany jest w leczeniu zaburzeń oddychania u wcześniaków. Jest to lek działający przyczynowo, gdyż jednym z głównych czynników w patogenezie RDS jest niedobór surfaktantu wynikający z wcześniactwa. Podanie dotchawicze surfaktantu u dziecka wentylowanego mechanicznie pozwala niejednokrotnie w ciągu minut, lub godzin, zmniejszyć nastawy respiratora. Niestety u niektórych pacjentów efekt leku jest przejściowy, a u innych może w ogóle nie wystąpić. Dotyczy to głównie noworodków, u których istnieją inne, oprócz samego wcześniactwa, czynniki wpływające na wystąpienie i nasilenie przebiegu RDS, takie jak: zakażenie wewnątrzmaciczne, niedotlenienie okołoporodowe. Czasem surfaktant podawany jest kilkakrotnie, w odstępach 12-24 godzinnych. Samo podanie leku nie jest procedurą pozbawioną możliwości powikłań. W trakcie podawania może dojść do zablokowania rurki intubacyjnej, albo jej przemieszczenia. Powikłania te mogą krótkotrwale, lecz w sposób istotny wpłynąć na wymianę

gazową. Nawet krótkotrwała destabilizacja dziecka może prowadzić do dalszych powikłań, takich jak wylew śródczaszkowy. Nierzadko obserwowana gwałtowana poprawa parametrów mechaniki oddychania po podaniu surfaktantu wymaga odpowiedniego dostosowania nastaw respiratora. Brak odpowiedniej reakcji może prowadzić do nadmiernego rozdęcia płuc i tak zwanych zespołów przecieku powietrza - rozedmy śródmiąższowej i odmy opłucnowej, które mogą prowadzić do nasilenia zaburzeń oddechowych, a nawet do zgonu.

Fakt wystąpienia chorób takich jak odma, posocznica, NEC, PDA, IVH może być także parametrem modelowania zjawisk związanych z zaburzeniami oddychania. Noworodki urodzone przedwcześnie, szczególne te, u których występują nasilone zaburzenia oddechowe, narażone są na szereg powikłań. Odma opłucnowa, czy posocznica mogą być bezpośrednią przyczyną zgonu. Wylew śródczaszkowy IVH może także doprowadzić do śmierci pacjenta lub trwałych następstw neurologicznych. Martwicze zapalenie jelit NEC może doprowadzić do posocznicy, perforacji jelit i znacznego pogorszenia stanu ogólnego dziecka. Obecność przetrwałego przewodu tętniczego PDA prowadzi do nasilenia zaburzeń oddechowych, utrwala zależność pacjenta od wspomagania oddychania z pomocą respiratora i prowadzi do przewlekłego powikłania w postaci dysplazji oskrzelowo-płucnej BPD.

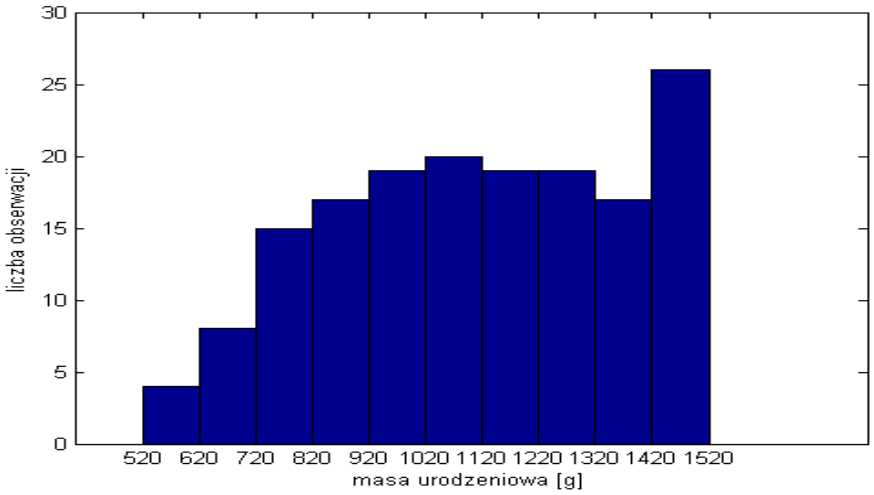
Fakt wystąpienia zgonu do końca 2 tygodnia życia dziecka jest parametrem modelowania. Pomimo dynamicznego rozwoju medycyny wieku noworodkowego, w tym głównie intensywnej opieki medycznej nad dziećmi urodzonymi przedwcześnie, zespół zaburzeń oddychania RDS związany z wcześniactwem stanowi wiodącą przyczynę śmiertelności i innych ciężkich powikłań w tej grupie pacjentów. Kluczowe znaczenie dla przeżywalności dzieci i uniknięcia ciężkich powikłań ma pierwsze 2 tygodnie życia. Jest to okres charakteryzujący się znaczną niestabilnością stanu ogólnego pacjentów, dużymi wahaniami parametrów życiowych i wysokim ryzykiem wystąpienia ciężkich powikłań. Czas trwania zaburzeń oddechowych może być zróżnicowany, zależny od stopnia dojrzałości dziecka przy urodzeniu oraz szeregu innych czynników. Brak możliwości opanowania zaburzeń związanych z wcześniactwem może prowadzić w tym okresie życia do zgonu przed upływem dwóch tygodni.

2.2 Dane dla metody AIS - predykcja wskaźnika pO_2/FiO_2

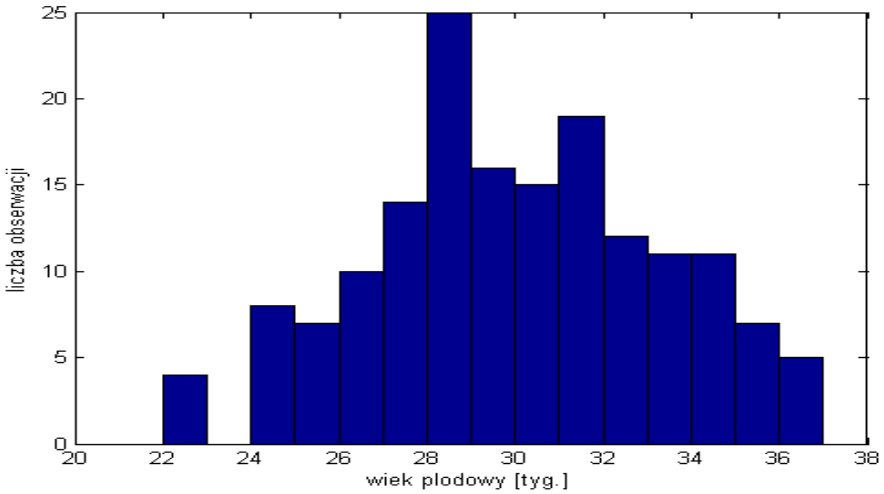
Do predykcji wskaźnika pO_2/FiO_2 z zastosowaniem AIS wykorzystano dane pacjentów z grupy drugiej, trzeciej i czwartej ze wsparciem oddechowym IMV/SIMV. Liczba pacjentów była równa 164 w tym 63 pacjentów wymagało podania surfaktantu. W tablicy 2.1 zamieszczono charakterystykę najważniejszych parametrów pacjentów. Parametry te są wykorzystywane w modelowaniu metodami sztucznych sieci immunologicznych.

Parametr	min	max	mean	std
masa urodzeniowa [g]	520	1500	1112	259.5
wiek płodowy [tyg.]	22	36	29.5	3.3

Tablica 2.1: Charakterystyka badanej grupy dla metody AIS



Rysunek 2.1: Histogram parametru: masa urodzeniowa dla metody AIS



Rysunek 2.2: Histogram parametru: wiek płodowy dla metody AIS

2.3 Dane dla metody SVM - ocena ryzyka wczesnego zgonu

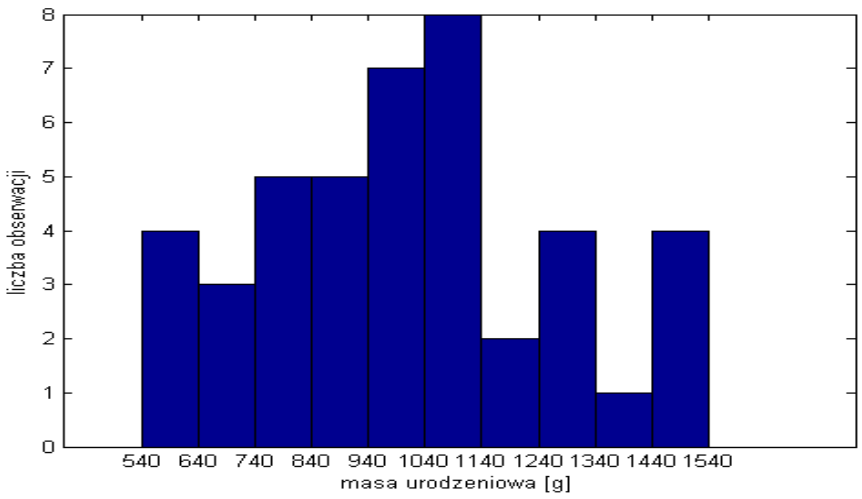
Do oceny ryzyka wczesnego zgonu (do końca 2 tyg. życia) z zastosowaniem metody SVM wykorzystano dane pacjentów ze wsparciem oddechowym IMV/SIMV, którzy w czasie oceny ryzyka zgonu należeli do grupy czwartej.

2.3.1 Pierwszy dzień hospitalizacji [24 godzina]

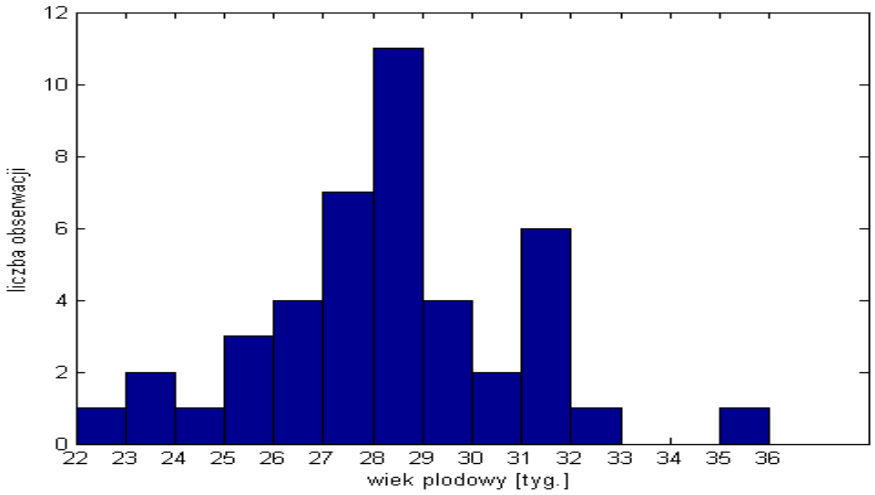
Pierwszy dzień hospitalizacji dotyczył 43 pacjentów w tym 24 pacjentów wymagało podania surfaktantu. W grupie tej wystąpiło 12 zgonów do 14 doby życia. W tablicy 2.2 zamieszczono charakterystykę najważniejszych parametrów pacjentów.

Parametr	min	max	mean	std
masa urodzeniowa [g]	540	1500	1001.2	261.6
wiek płodowy [tyg.]	22	36	28.37	2.7
pO_2/FiO_2 [mmHg]	38.67	99.90	78.73	16.12
HCO_3 [mmol/l]	14.97	26.89	22.18	2.26
pH	7.05	7.40	7.24	0.09
pCO_2 [mmHg]	30.84	93.44	55.07	14.13

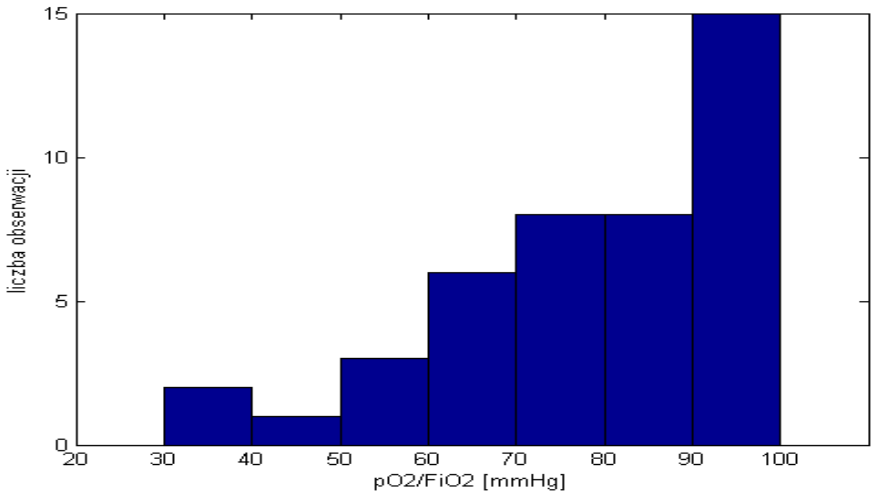
Tablica 2.2: Charakterystyka badanej grupy dla metody SVM - 24 godz.



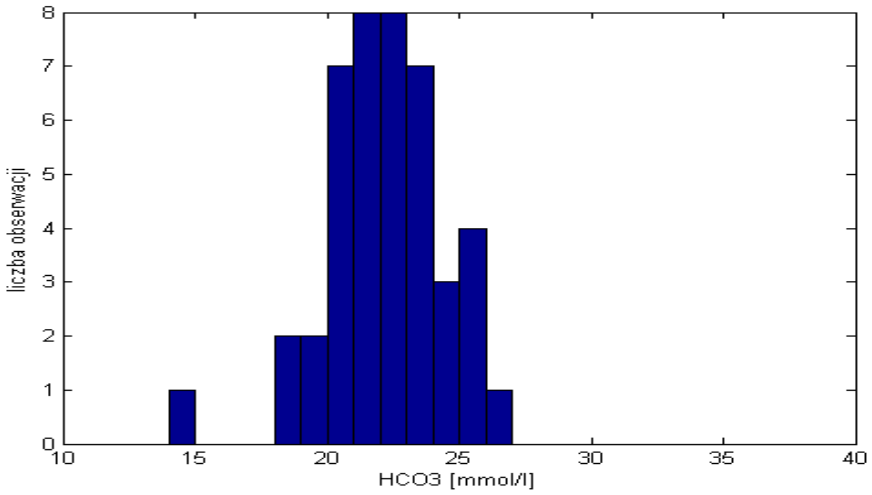
Rysunek 2.3: Histogram parametru: masa urodzeniowa dla metody SVM - 24 godz.



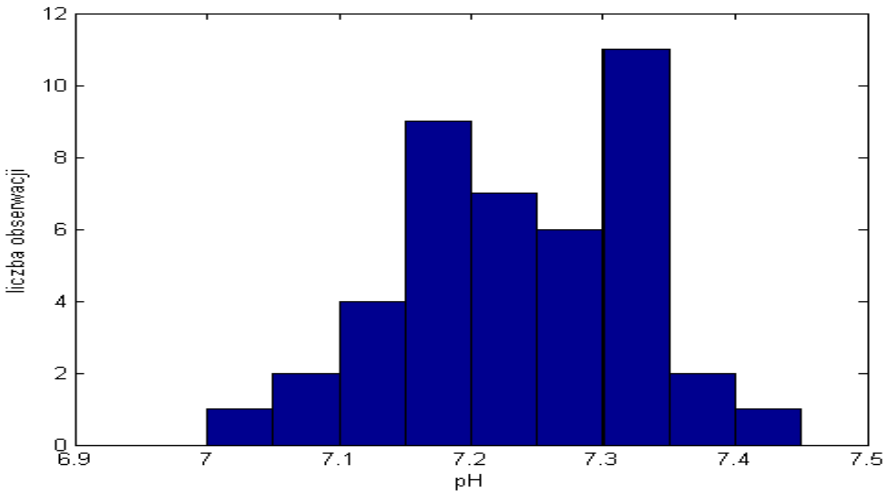
Rysunek 2.4: Histogram parametru: wiek płodowy dla metody SVM - 24 godz.



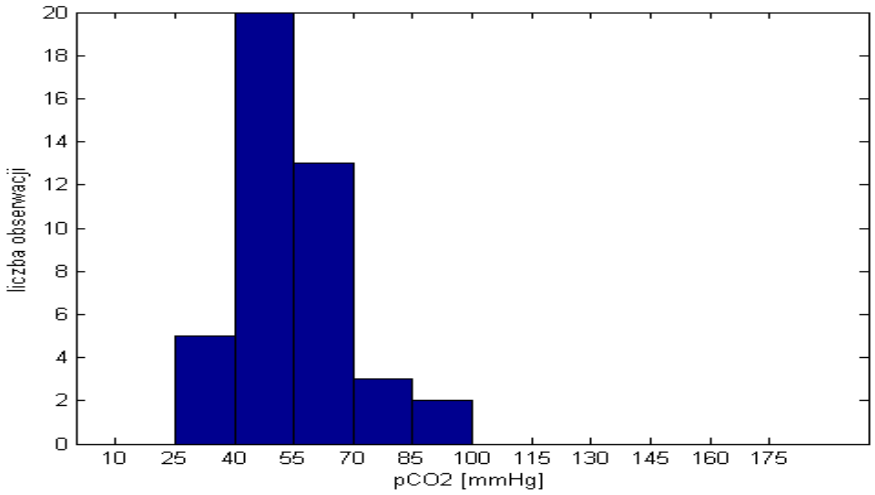
Rysunek 2.5: Histogram parametru: pO_2/FiO_2 dla metody SVM - 24 godz.



Rysunek 2.6: Histogram parametru: HCO_3 dla metody SVM - 24 godz.



Rysunek 2.7: Histogram parametru: pH dla metody SVM - 24 godz.



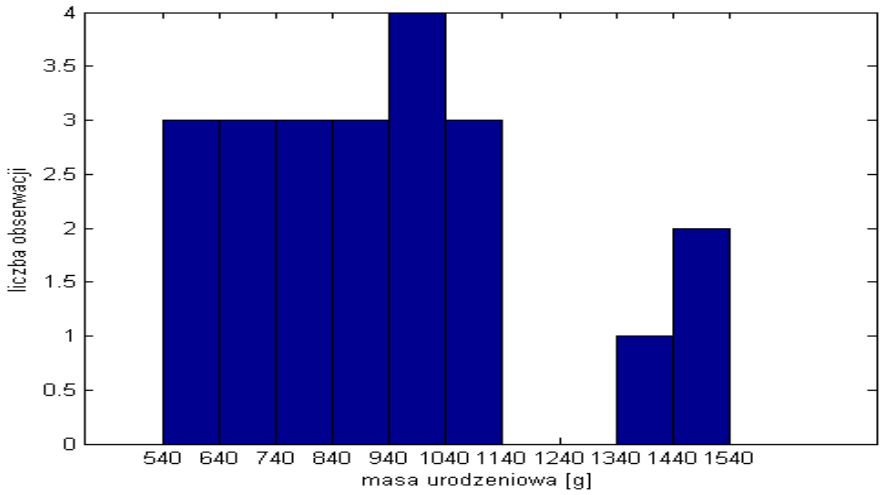
Rysunek 2.8: Histogram parametru: pCO_2 dla metody SVM - 24 godz.

2.3.2 Drugi dzień hospitalizacji [48 godzina]

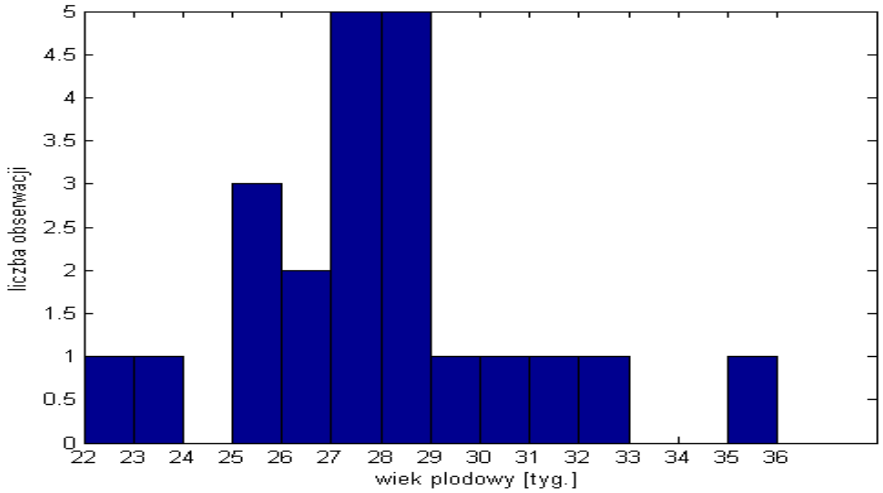
Drugi dzień hospitalizacji dotyczył 22 pacjentów w tym 10 pacjentów wymagało podania surfaktantu. W grupie tej wystąpiło 8 zgonów do 14 doby życia. W tablicy 2.3 zamieszczono charakterystykę najważniejszych parametrów pacjentów.

Parametr	min	max	mean	std
masa urodzeniowa [g]	560	1470	926.4	268.2
wiek płodowy [tyg.]	22	36	28	2.9
pO_2/FiO_2 [mmHg]	34.74	98.98	71.98	19.43
HCO_3 [mmol/l]	15.70	31.43	22.25	3.24
pH	7.08	7.36	7.22	0.08
pCO_2 [mmHg]	36.20	130.31	58.76	19.94

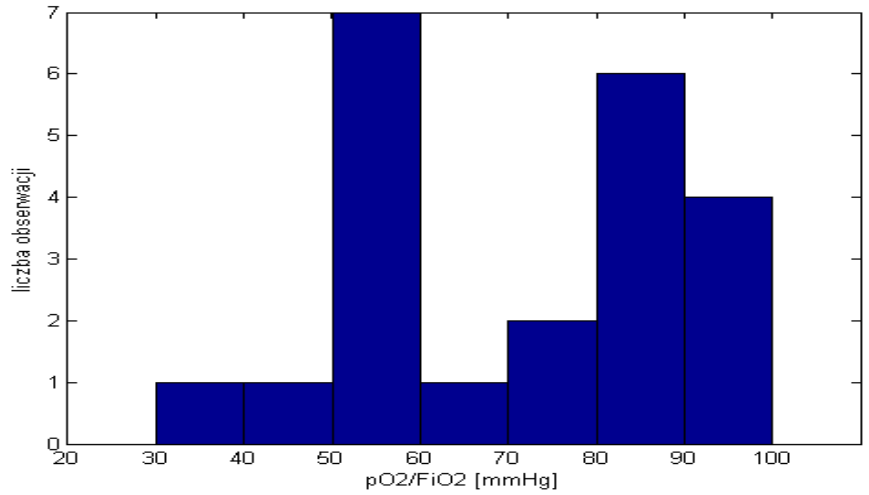
Tablica 2.3: Charakterystyka badanej grupy dla metody SVM - 48 godz.



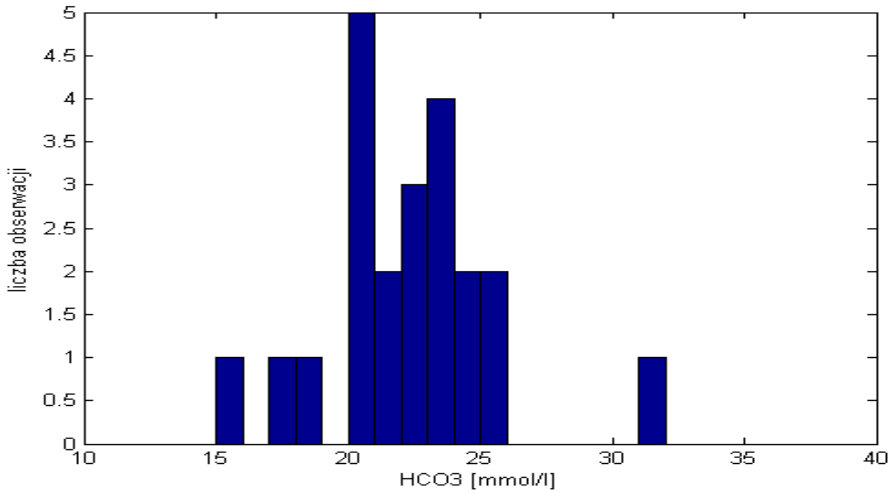
Rysunek 2.9: Histogram parametru: masa urodzeniowa dla metody SVM - 48 godz.



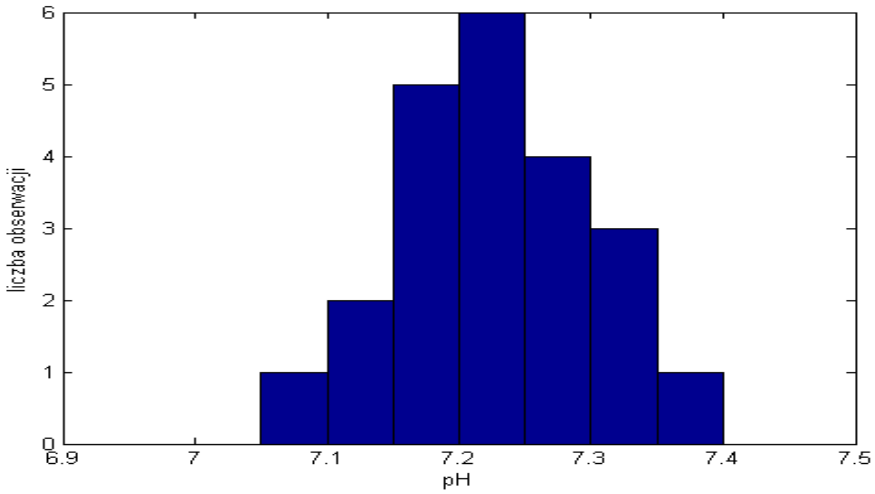
Rysunek 2.10: Histogram parametru: wiek płodowy dla metody SVM - 48 godz.



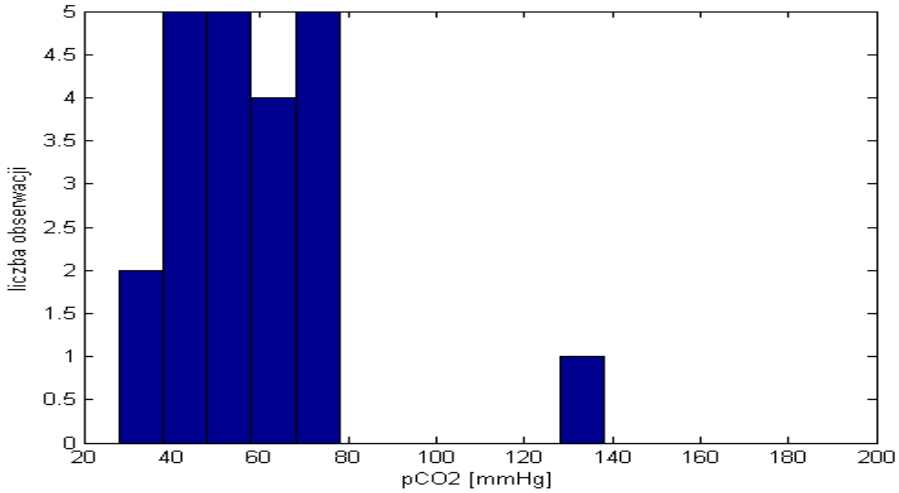
Rysunek 2.11: Histogram parametru: pO_2/FiO_2 dla metody SVM - 48 godz.



Rysunek 2.12: Histogram parametru: HCO_3 dla metody SVM - 48 godz.



Rysunek 2.13: Histogram parametru: pH dla metody SVM - 48 godz.



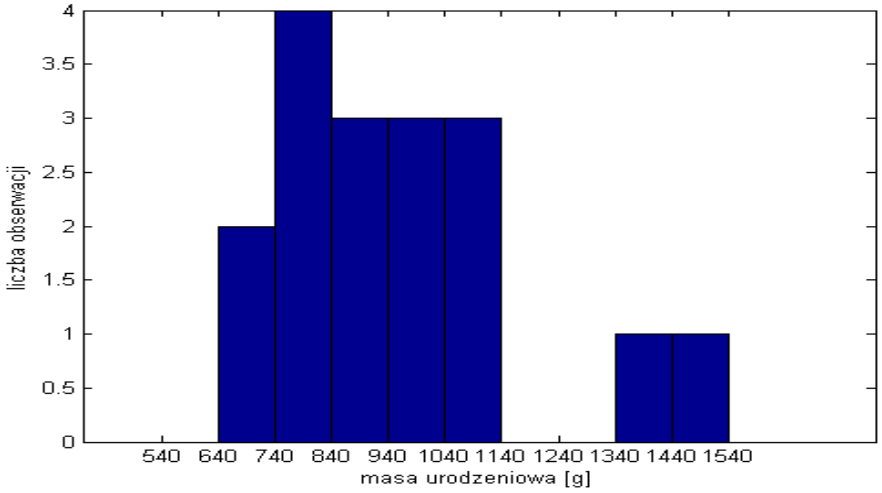
Rysunek 2.14: Histogram parametru: pCO_2 dla metody SVM - 48 godz.

2.3.3 Trzeci dzień hospitalizacji [72 godzina]

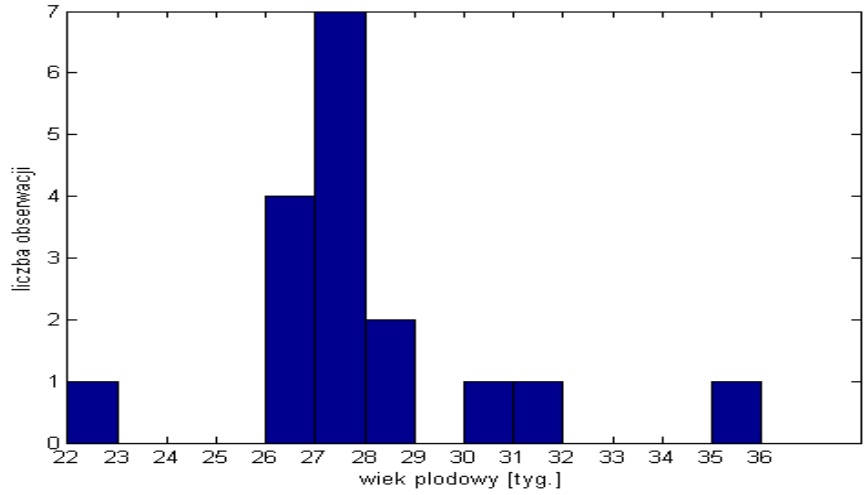
Trzeci dzień hospitalizacji dotyczył 17 pacjentów w tym 4 pacjentów wymagało podania surfaktantu. W grupie tej wystąpiło 7 zgonów do 14 doby życia. W tablicy 2.4 zamieszczono charakterystykę najważniejszych parametrów pacjentów.

Parametr	min	max	mean	std
masa urodzeniowa [g]	650	1450	948.2	230.6
wiek płodowy [tyg.]	22	36	28	2.9
pO_2/FiO_2 [mmHg]	31.75	97.26	62.84	18.95
HCO_3 [mmol/l]	18.25	35.82	23.06	4.66
pH	6.93	7.41	7.22	0.11
pCO_2 [mmHg]	29.50	170.88	66.09	31.31

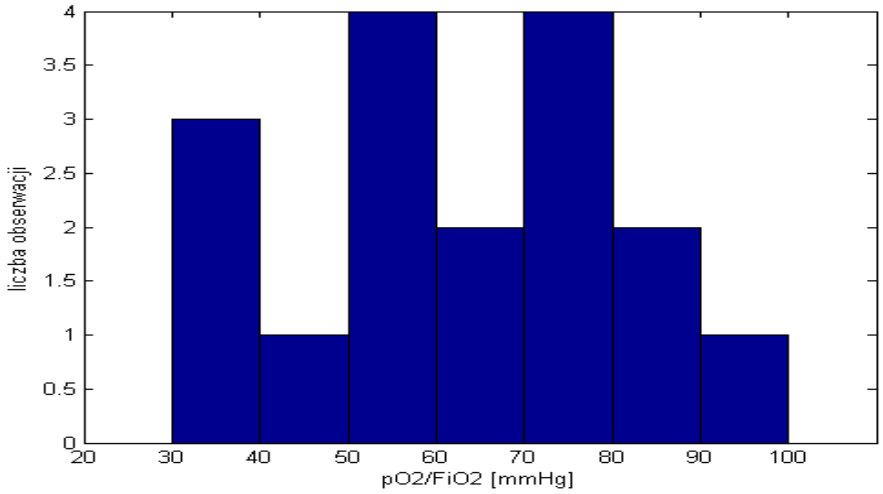
Tablica 2.4: Charakterystyka badanej grupy dla metody SVM - 72 godz.



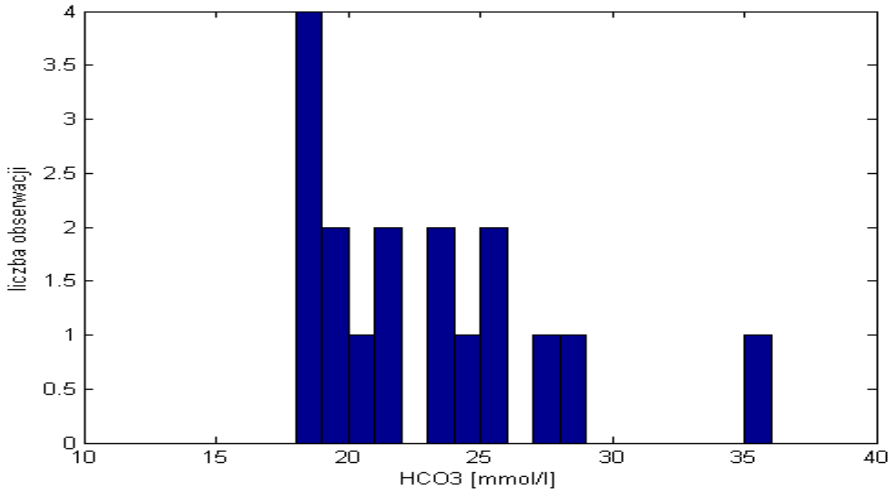
Rysunek 2.15: Histogram parametru: masa urodzeniowa dla metody SVM - 72 godz.



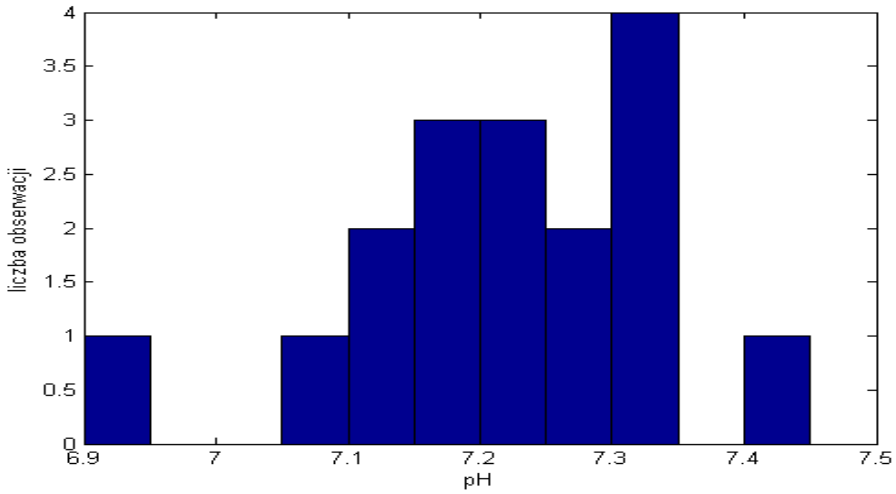
Rysunek 2.16: Histogram parametru: wiek płodowy dla metody SVM - 72 godz.



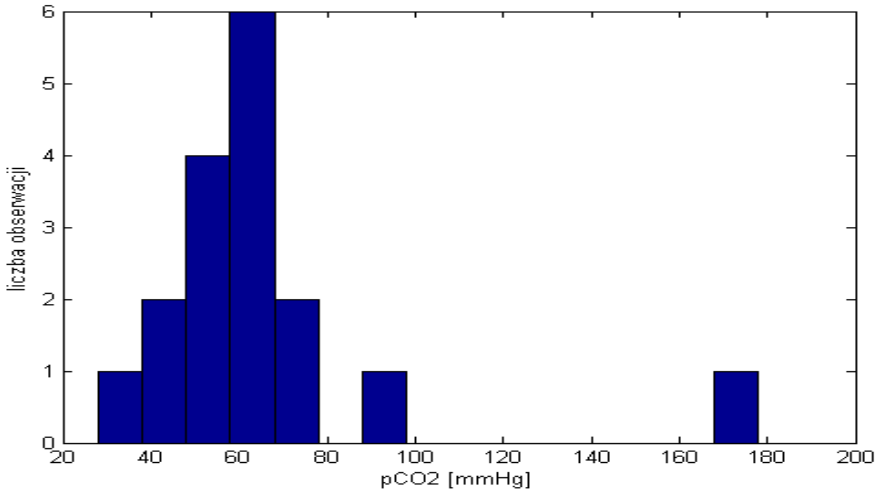
Rysunek 2.17: Histogram parametru: pO_2/FiO_2 dla metody SVM - 72 godz.



Rysunek 2.18: Histogram parametru: HCO_3 dla metody SVM - 72 godz.



Rysunek 2.19: Histogram parametru: pH dla metody SVM - 72 godz.



Rysunek 2.20: Histogram parametru: pCO_2 dla metody SVM - 72 godz.

Rozdział 3

Sztuczne sieci immunologiczne w modelowaniu parametrów medycznych

Sztuczny system immunologiczny (ang. Artificial Immune System AIS) jest definiowany jako struktura oraz mechanizm funkcjonowania bazujący na systemie odpornościowym kręgowców. W rozdziale zastosowano algorytm AIS do generowania przewidywanych stanów zdrowia dziecka. Zadaniem algorytmu jest określenie wybranych parametrów opisujących stan zdrowia w najbliższej przyszłości określonej w godzinach na podstawie danych zgromadzonych w bazie danych. Zastosowano dwa różne algorytmy AIS, przy założeniu dostatecznie licznej reprezentacji liczbowej za pomocą, której można opisać stany chorobowe. Te reprezentacje liczbowe są zgromadzone w bazie danych. Liczebność tych danych warunkuje otrzymanie rezultatów z zadowalającą dokładnością.

3.1 Reprezentacja przeciwciał i antygenów

W początkowym okresie stosowania algorytmów immunologicznych zakładano, że przeciwciała i antygeny są modelowane przy pomocy łańcuchów binarnych. Obecnie jest stosowana reprezentacja, w której antygen lub przeciwciało są przedstawiane za pomocą wektorów liczb rzeczywistych. Zastosowanie algorytmów immunologicznych w tej pracy wymaga, aby relacja podobieństwa antygenu do przeciwciała była wartością numeryczną. Można przyjąć, że przypadek medyczny $\mathbf{m}$ jest reprezentowany w przestrzeni liczb rzeczywistych w postaci wektora:

$$\mathbf{m} =< m_1, m_2, ..., m_N >, \tag{3.1}$$

który można uważać jako punkt w przestrzeni N-wymiarowej. Podobieństwo pomiędzy antygenem, a przeciwciałem można interpretować jako odległość pomiędzy dwoma punktami w przestrzeni. Wartość określająca stopień dopasowania, inaczej powinowactwa, może być obliczona jako odległość euklidesowa

według wzoru 3.2

$$D = \sqrt{\sum_{i=1}^{i=N}(Ab_i - Ag_i)^2}. \tag{3.2}$$

Współrzędne przeciwciała (antybody) przedstawia

wektor$< Ab_1, Ab_2, ..., Ab_N >$ natomiast wektor $< Ag_1, Ag_2, ..., Ag_N >$ określa antygen.

W ogólnym przypadku, stopień powinowactwa można mierzyć korzystając z metryki Minkowskiego z indeksem r

$$D = (\sum_{i=1}^{i=N} |Ab_i - Ag_i|^r)^{\frac{1}{r}}, r \geq 1. \tag{3.3}$$

Przypadek, gdy $r = 1$ odległość pomiędzy dwoma punktami obliczamy przy użyciu *metryki ulic*, albo metryki manhattan

$$D = \sum_{i=1}^{i=N} |(Ab_i - Ag_i)|. \tag{3.4}$$

Innym modelem obliczania odległości jest iloczyn skalarny nieujemnych liczb rzeczywistych

$$D = Ab \cdot Ag^T = \sum_{i=1}^{i=N}(Ab_i \cdot Ag_i). \tag{3.5}$$

3.2 Algorytm Sztucznej Sieci Immunologicznej

Algorytm sztucznej sieci immunologicznej jest metodą, która bazuje na mechanizmach występujących podczas odpowiedzi immunologicznej organizmu w sytuacji, kiedy antygen zostanie zidentyfikowany przez system odpornościowy. Jest to algorytm selekcji pozytywnej nazywany również algorytmem selekcji klonalnej [4, 2, 6, 7, 5, 32, 28].

Algorytm wykorzystuje mechanizm związany z limfocytami typu B oraz mechanizm odpowiedzi komórek B, które zostaną aktywowane obecnością antygenu. Gdy przeciwciało, które jest dołączone do komórki zidentyfikuje antygen wówczas nastąpi przekazanie sygnału do wnętrza komórki. Sygnał ten zapoczątkuje proces podziału i różnicowania się aktywowanej komórki. Początkowo generowany jest zbiór **M** . Zbiór **M** zawiera przeciwciała. Zbiór **P** gromadzi wzorce, a posługując się przyjętym wcześniej nazewnictwem, zbiór ten zawiera antygeny. Następnym krokiem realizowanego algorytmu jest prezentacja antygenu wszystkim przeciwciałom i obliczenie wartości powinowactwa. W tym kroku algorytm określa według obliczonej wartości powinowactwa ustaloną liczbę N_1 komórek o największej wartości powinowactwa. Wybrane przeciwciała podawane są procesowi klonowania. Podczas procesu klonowania obowiązuje zasada, że przeciwciało, które zidentyfikowało określony antygen generuje największą liczbę kopii w porównaniu do przeciwciał o mniejszej wartości powinowactwa. Drugi krok algorytmu

realizuje proces mutacji nowo utworzonych kopii przeciwciał. Przeciwciała mutowane są w ten sposób, że przeciwciało o największej wartości powinowactwa będzie miało najniższą wartość współczynnika mutacji, czyli zostanie zmodyfikowane w mniejszym stopniu. W trzecim kroku, realizowana jest operacja dodawania nowoutworzonych przeciwciał do zbioru **M** zawierającego przeciwciała. Algorytm jest zatrzymywany wtedy, kiedy zostaną spełnione kryteria stopu. Kryterium to jest związane z liczebnością poprawnie rozpoznanych wzorców **P**, albo zadaną wartością błędu. Algorytm rozpoczyna działanie od wygenerowania zbioru **M** przeciwciał w sposób losowy. Następnie dla każdego antygenu należącego do zbioru **P** , prezentowane są przeciwciała należące do zbioru **M**, a następnie wyznaczana jest wartość powinowactwa dla każdego elementu zbioru **M**.

Algorytm składa się z czterech kroków:

krok 1. Wybierz N_1 przeciwciał o największej wartości powinowactwa ze zbioru **M** i utwórz ich kopie proporcjonalnie do wartości powinowactwa do antygenu. Przeciwciało o największej wartości powinowactwa posiada największą liczbę kopii.

krok 2. Poddaj procesowi mutacji wszystkie nowo utworzone kopie przeciwciał ze wskaźnikiem mutacji odwrotnie proporcjonalnym do wartości powinowactwa. Duża wartość powinowactwa przeciwciała do antygenu implikuje niską wartość wskaźnika mutacji.

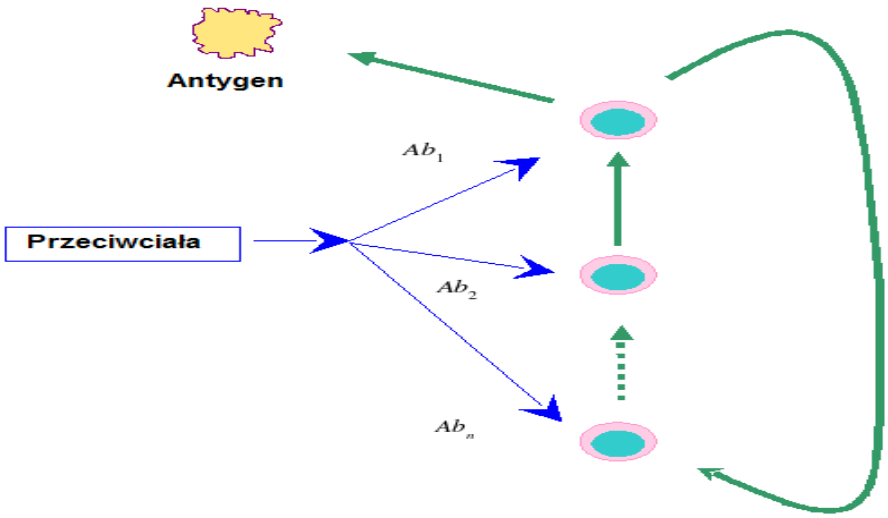
krok 3. Dodaj wszystkie zmodyfikowane w poprzednim kroku przeciwciała do zbioru.

krok 4. Powtarzaj kroki 1,2,3 do momentu spełnienia zadanych kryteriów stopu.

Mechanizm funkcjonowania algorytmu selekcji klonalnej jest bardzo podobny do klasycznego algorytmu genetycznego. Jedną z różnic, która ma decydujące znaczenie dla jego własności jest to, że w przedstawionym algorytmie nie występuje operacja krzyżowania. Brak operacji krzyżowania powoduje, że populacja potencjalnych rozwiązań zachowuje swoją różnorodność. Zachowanie różnorodności populacji ma decydujące znaczenie w przypadku optymalizacji funkcji wielomodalnych.

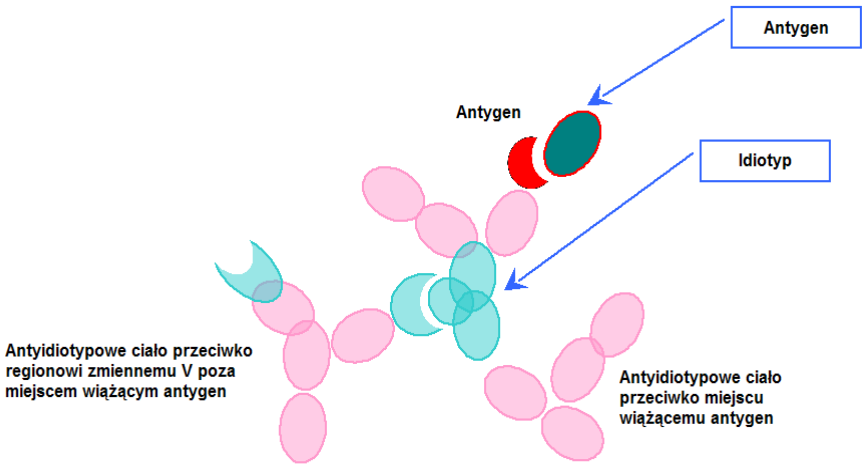
3.2.1 Paradygmat sieci antyidiotypowej

Idea sztucznych sieci immunologicznej została zaproponowana w 1974 roku [2]. Podstawowym założeniem, które częściowo wynika z obserwacji systemu immunologicznego, jest hipoteza dotycząca limfocytów. Zaobserwowano, że w sytuacji, kiedy system immunologiczny nie jest zainfekowany żadnym antygenem, limfocyty ciągle wykazują dużą aktywność. Właśnie ta obserwacja skłoniła do postawienia hipotezy stanowiącej, że podczas funkcjonowania systemu immunologicznego niektóre przeciwciała są zdolne do rozpoznawania innych przeciwciał. To założenie sugeruje, że system immunologiczny buduje zbiór komórek. Zadaniem każdej z tych komórek jest kontrola aktywności komórek sąsiednich. Rysunek 3.1



Rysunek 3.1: Formowanie pętli antyidiotypowej

Sieć immunologiczną można zdefiniować jako zbiór komórek, które tworzą między sobą wiązania antyidiotypowe albo wiązania typu przeciwciało-antygen, w tym przypadku wiązania idiotypowe. Rysunek 3.2.



Rysunek 3.2: Przeciwciała idiotypowe i antyidiotypowe

Odpowiedź immunologiczna tak zdefiniowanego systemu może mieć charakter zarówno pozytywny jak też negatywny, ponieważ system może rozpoznawać nie tylko antygen, ale także inne przeciwciała, które stanowią podstawę sieci immunologicznej. Pozytywna odpowiedź systemu wystąpi wówczas, kiedy antygen zostanie zidentyfikowany. Rezultatem identyfikacji antygenu jest aktywacja odpowiedniej komórki typu B, co zainicjuje proces jej podziału, różnicowania, powstawania komórek plazmatycznych i uwalniania przeciwciał. Negatywna odpowiedź prowadzi do supresji i tolerancji systemu.

System immunologiczny jest stymulowany przez antygen. Mechanizm ten może być opisany w następujący sposób. Antygen jest rozpoznawany przez zbiór przeciwciał, które występują na powierzchni komórek typu B i są jej receptorami. Siła wiązania antygen - przeciwciało jest proporcjonalna do wartości obliczonego powinowactwa. Wykorzystując cechę proporcjonalności do powinowactwa zostaje utworzony zbiór komórek typu B, które są zdolne wytworzyć wiązanie z antygenem. Powstaje zbiór idiotypów danego antygenu, który stanowi wewnętrzny obraz antygenu. Ten zbiór idiotypów może być rozpoznawany przez zbiór innych komórek B. Komórki te są zdolne do wytworzenia wiązania antyidiotypowego. Efekt stymulacji występuje wówczas, kiedy antygen jest rozpoznawany przez przeciwciała. Z efektem tolerancji immunologicznej mamy do czynienia wówczas, kiedy te przeciwciała, które były zdolne do rozpoznania danego antygenu są rozpoznawane i wiązane przez wytworzenie wiązania antyidiotypowego z innymi przeciwciałami tworzącymi sieć immunologiczną. Przedstawione własności sieci antyidiotypowej zostały wykorzystane w realizacji mechanizmu uczenia sieci immunologicznej.

3.2.2 Mechanizm uczenia sieci immunologicznej

Zaprezentowany algorytm uczenia sztucznej sieci immunologicznej został przedstawiony w pracy [2]. Sztuczna sieć immunologiczna wykorzystuje mechanizm wiązań idiotypowych i antyidiotypowych oparty na interakcjach związanych z limfocytami typu B. Pierwszym krokiem algorytmu jest utworzenie populacji antygenów **Ag**. Zbiór antygenów jest zbiorem uczącym, który zawiera wzorce. Drugim istotnym krokiem algorytmu jest generowanie populacji komórek B. W opisie algorytmu rola komórek B została zredukowana do funkcji przeciwciała. Przeciwciała pełnią funkcje receptorów dla antygenów. Populacja komórek B generowana jest w sposób losowy. Wielu autorów [2, 32] zaleca dokonać podziału danych uczących na dwa rozłączne zbiory i jeden z tych zbiorów traktować jako populację wykorzystaną w procesie uczenia komórek B dla algorytmu. Kolejnym krokiem jest etap uczenia sieci immunologicznej. Dla każdego antygenu, który należy do zbioru **Ag** przeprowadzane są obliczenia opisane w krokach od 1 do 10 algorytmu uczenia sieci idiotypowej. W kroku pierwszym obliczana jest wartość powinowactwa j-tego antygenu do wszystkich przeciwciał zawartych w zbiorze **Ab**. Element macierzy $D_{i,j}$ określa wartość odległości. Mniejsza wartość odległości pomiędzy przeciwciałem a antygenem to większe powinowactwo między nimi. Kolejnym krokiem algorytmu jest utworzenie n-elementowego zbioru przeciwciał o najwyższym powinowactwie do j-tego antygenu. Dla n-elementowego

zbioru przeciwciał realizowany jest proces klonowania. Dla procesu klonowania obowiązuje zasada, że przeciwciało o największym powinowactwie posiada największą liczbę swoich kopii. Nowo utworzone przeciwciała są mutowane według podobnej reguły. Przeciwciała o najwyższej wartości powinowactwa mutowane są wskaźnikiem mutacji o najmniejszej wartości. W rezultacie w kroku 4 algorytmu powstaje nowy zbiór przeciwciał oznaczony symbolem $\mathbf{C}^*$. W krokach 5 i 6 algorytmu, realizowana jest procedura wyboru subpopulacji nowoutworzonych przeciwciał ze zbioru $\mathbf{C}^*$ **.** Celem jest utworzenie z tych komórek, komórek pamięciowych, które zostaną zapamiętane w zbiorze $\mathbf{M}$.

3.2.3 Algorytm AIS nr 1

$\forall Ag_j \in \mathbf{Ag}$ wykonaj kroki od 1 do 10:

krok 1. Wyznacz powinowactwa pomiędzy j-tym antygenem Ag_j $f_{i,j}$ $dla\, i = 1, ..., N$ do wszystkich przeciwciał Ab_i, $f_{i,j} = 1/D_{i,j}$, dla $i = 1, ..., N$;

krok 2. Utwórz zbiór Ab_n, który zawiera $n <= N$ przeciwciał o najwyższej wartości powinowactwa według zadanego kryterium;

krok 3. n wybranych przeciwciał poddawane są procesowi klonowania proporcjonalnie do ich powinowactwa do antygenów $f_{i,j}$ oraz następuje wygenerowanie zbioru klonów $\mathbf{C}$ o najwyższej wartości powinowactwa według zadanego kryterium;

krok 4. Elementy zbioru $\mathbf{C}$ poddawane są mutacji i w rezultacie generowany jest zbiór $\mathbf{C}^*$, prawdopodobieństwo mutacji jest odwrotnie proporcjonalne do powinowactwa przeciwciał rodzicielskich do j-tego antygenu Ag_j $f_{i,j}$. Duże powinowactwo implikuje małe prawdopodobieństwo mutacji;

krok 5. Wyznacz powinowactwa $d_{k,j} = 1/D_{k,j}$ pomiędzy Ag_j dla każdego elementu k zbioru $\mathbf{C}^*$;

krok 6. Ze zbioru $\mathbf{C}^*$, należy wybrać $\xi[\%]$ przeciwciał, które mają największy wskaźnik powinowactwa $d_{k,j}$ i utworzyć z nich M_j komórek pamięciowych;

krok 7. Śmierć komórek: Eliminacja wszystkich komórek pamięciowych ze zbioru M_j dla których wartość powinowactwa spełnia relacje $D_{k,j} > \sigma_d$;

krok 8. Wyznacz powinowactwa pomiędzy klonowanymi przeciwciałami $s_{i,k} = |Ab_i - Ab_j|$;

krok 9. Supresja: Eliminacja tych klonów ze zbioru M_j, dla których spełniona jest relacja $s_{i,k} < \sigma_s$, gdzie σ_s jest zadaną arbitralnie wartością;

krok 10. Utworzenie zbioru ze wszystkich przeciwciał, które są zapisane w zbiorach M_j

Po wykonaniu obliczeń w krokach od 1 do 10 dla każdego antygenu należącego do zbioru $\mathbf{Ag}$ należy wykonać:

- Powinowactwo: wyznaczenie powinowactwa pomiędzy wszystkimi przeciwciałami pamięciowymi Ab_m:
- Supresja: eliminacja tych przeciwciał, które spełniają relacje: $s_{i,k} < \sigma_s$:
- Generacja: utworzenie zbioru przeciwciał $\mathbf{Ab} \leftarrow [Ab_m; Ab_d]$,

gdzie:

Ab : Dostępny zbiór przeciwciał,

Ab_m: Zbiór przeciwciał, które są komórkami pamięciowymi ,

Ab_d: Nowo utworzone przeciwciała,

Ag: Populacja antygenów,

f_j: Wektor zawierający wartość powinowactwa dla wszystkich przeciwciał Ab_i, $i = 1, ..., N$ w relacji do j-tego antygenu Ag_j. W tym przypadku powinowactwo jest odwrotnie proporcjonalnie do odległości,

S: Macierz podobieństwa pomiędzy każdą z par przeciwciał Ab_i i Ab_j, $s_{i,j}$; $i, j = 1, ..., N$,

C: Zbiór klonów przeciwciał o największym powinowactwie wygenerowany na bazie elementów zbioru **Ab**,

$\mathbf{C}^*$: Zbiór przeciwciał, które zostały utworzone z elementów zbioru **C**, które zostały poddane procesowi mutacji,

d_j: Wektor zawierający wartości powinowactwa pomiędzy elementem zbioru $\mathbf{C}^*$ i każdym elementem zbioru Ag_j,

ξ: Parametr określający, jaki procent populacji przeciwciał będzie poddawane procesowi klonowania w procesie aktywacji antygenem,

M_j: Wektor komórek pamięciowych dla antygenu Ag_j (pozostałe po procesie supresji), są to komórki pamięciowe utworzone z j-tego przeciwciała Ag_j,

σ_d: liczba określająca próg usuwania komórek,

σ_s: liczba określająca próg supresji.

Podobnie jak w naturalnym systemie immunologicznym, również w sztucznym systemie, komórki B, które po zmutowaniu w sposób znaczący różnią się między sobą zostaną usunięte. W ten sposób system odpornościowy zabezpiecza się przed powstaniem komórek, które byłyby komórkami autoreaktywnymi. Podobna operacja jest realizowana w algorytmie i pojęcie znaczącej różnicy pomiędzy komórkami jest określona wartością parametru σ_d. Ostatnim elementem algorytmu, który jest realizowany dla każdego antygenu jest operacja usunięcia tych komórek pamięciowych, które są do siebie bardzo podobne, co w przypadku naturalnego systemu immunologicznego należy interpretować jako hamowanie odpowiedzi immunologicznej systemu odpornościowego. Wartością decydującą o przebiegu procesu jest wartość parametru σ_s. Wartość parametru σ_s ma decydujący wpływ na zdolność do generalizacji sieci immunologicznej. Rezultatem algorytmu jest zbiór **Ab**, który zawiera zbiór przeciwciał utworzonych w procesie uczenia sieci immunologicznej. Kryteriami zatrzymania algorytmu może być liczba iteracji algorytmu lub wartość błędu związanego z identyfikacją wzorców zawartych w zbiorze uczącym **Ag**.

3.2.4 Przykład realizacji algorytmu AIS nr 1

Przykład przeprowadzono dla jednego antygenu - kroki od 1 do 10. Dane w tym przykładzie są oryginalne, znormalizowane i pochodzą z bazy danych.

Tablica 3.1 przedstawia znormalizowany zbiór do nauki AIS - Ags.

	pO_2/FiO_2												
Nr pacj	0 godz.	6 godz.	12 godz.	18 godz.	24 godz.	30 godz.	36 godz.	42 godz.	48 godz.	54 godz.	60 godz.	66 godz.	72 godz.
1	0,00	0,16	0,34	0,53	0,74	0,92	1,00	0,99	0,91	0,83	0,76	0,69	0,64
2	0,00	0,03	0,05	0,07	0,10	0,13	0,19	0,28	0,41	0,58	0,76	0,92	1,00
3	0,00	0,08	0,15	0,23	0,34	0,47	0,63	0,79	0,92	0,99	1,00	0,97	0,90
4	1,00	0,94	0,86	0,76	0,66	0,55	0,41	0,22	0,06	0,00	0,08	0,27	0,54
5	0,00	0,04	0,08	0,12	0,17	0,23	0,31	0,40	0,51	0,63	0,78	0,91	1,00
6	0,76	0,43	0,19	0,05	0,00	0,02	0,11	0,23	0,39	0,57	0,73	0,87	1,00
7	0,32	0,21	0,12	0,05	0,00	0,00	0,05	0,15	0,31	0,50	0,71	0,89	1,00
8	0,00	0,14	0,32	0,54	0,75	0,90	0,99	1,00	0,96	0,90	0,84	0,81	0,82
9	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
10	0,57	0,77	0,92	1,00	1,00	0,92	0,74	0,50	0,29	0,15	0,08	0,04	0,00

Tablica 3.1: Zbiór **Ags** do nauki AIS

Parametry wykorzystane w procesie uczenia algorytmu AIS w przykładzie:
$pa = 1$, $ma = 13$, $tp = 0.5$, $qi = 0.3$, $gen = 1$, $N = 20$, $n = 4$, $ts = 0.005$,
gdzie: pa liczba parametrów gazometrii krwi (pO_2/FiO_2), ma liczba pomiarów dla jednego parametru krwi, n liczba najlepiej dopasowanych komórek w odniesieniu do każdego antygenu (ang. no. of best-matching cells taken for each Ag (Selection)), N liczba klonów (ang. clone number multiplier), gen maksymalna liczba generacji (ang. maximum number of generations), qi procentowa zawartość klonów, które są ponownie poddane procesowi selekcji (ang. percentile amount of clones to be Re-selected), ts próg supresji (ang. suppression threshold (default: 0.001)), tp próg usuwania komórek (ang. pruning threshold).

Wywołanie funkcji ainet_sun w środowisku Matlab.
Cały kod programu znajduje się w dodatku A.

$$[M1, D1] = ainet\_sun(Ags, ts, n, N, gen, qi, tp, pa, ma);$$

W funkcji ainet_sun następuje losowy podział zbioru Ags na dwa zbiory: **Ag** zbiór antygenów (tablica 3.2) oraz **Ab** zbiór przeciwciał (tablica 3.3).

	pO_2/FiO_2												
Nr pacj	0 godz.	6 godz.	12 godz.	18 godz.	24 godz.	30 godz.	36 godz.	42 godz.	48 godz.	54 godz.	60 godz.	66 godz.	72 godz.
1	0,00	0,08	0,15	0,23	0,34	0,47	0,63	0,79	0,92	0,99	1,00	0,97	0,90
2	0,76	0,43	0,19	0,05	0,00	0,02	0,11	0,23	0,39	0,57	0,73	0,87	1,00
3	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
4	0,00	0,04	0,08	0,12	0,17	0,23	0,31	0,40	0,51	0,63	0,78	0,91	1,00
5	0,00	0,16	0,34	0,53	0,74	0,92	1,00	0,99	0,91	0,83	0,76	0,69	0,64

Tablica 3.2: Zbiór **Ag**

	pO_2/FiO_2												
Nr pacj	0 godz.	6 godz.	12 godz.	18 godz.	24 godz.	30 godz.	36 godz.	42 godz.	48 godz.	54 godz.	60 godz.	66 godz.	72 godz.
1	0,00	0,14	0,32	0,54	0,75	0,90	0,99	1,00	0,96	0,90	0,84	0,81	0,82
2	0,00	0,03	0,05	0,07	0,10	0,13	0,19	0,28	0,41	0,58	0,76	0,92	1,00
3	0,57	0,77	0,92	1,00	1,00	0,92	0,74	0,50	0,29	0,15	0,08	0,04	0,00
4	0,32	0,21	0,12	0,05	0,00	0,00	0,05	0,15	0,31	0,50	0,71	0,89	1,00
5	1,00	0,94	0,86	0,76	0,66	0,55	0,41	0,22	0,06	0,00	0,08	0,27	0,54

Tablica 3.3: Zbiór **Ab**

krok 1. Wyznacz powinowactwa pomiędzy j-tym antygenem Ag_j $f_{i,j}$
$i = 1, ..., N$ do wszystkich przeciwciał Ab_i, $f_{i,j} = 1/D_{i,j}$, dla $i = 1, ..., N$;
$\mathbf{Ag} - \mathbf{Ab}$ powinowactwo (ang. Affinity (Match-Function: Euclidian distance))
D wektor odległości (ang. distance matrix), $k = 0$ (dla jednego parametru)

Wywołanie funkcji dist w środowisku Matlab:

$D = dist(Ab(:, (k * ma + 1) : (k + 1) * ma), Ag(vet(i), (k * ma + 1) :$
$(k + 1) * ma)')$;

Dla tego przykładu wybrano odległość euklidesową pomiędzy pierwszym antygenem czyli Ag_1 (numer pacjenta 1) a całą macierzą przeciwciał Ab_i.

Obliczenie odległości: $D = Ab_i - Ag_j$ dla $i = 1, ..., 5; j = 1$ generuje wektor D.

0,33	1,50	2,23	1,79	2,41

Tablica 3.4: Wektor D

Najmniejsza odległość (tablica 3.4) występuje pomiędzy Ag_1 a Ab_1 oznacza to największe powinowactwo.

krok 2. Utwórz zbiór Ab_m, który zawiera n przeciwciał o najwyższej wartości powinowactwa na bazie części kodu programu:

$[Dn, I] = sort(norma(D))$;
$Nc = floor(N - Dn(1 : n, :) * N)$;

Otrzymano posortowany i znormalizowany wektor odległości Dn.

0	0,56	0,70	0,91	1

Tablica 3.5: Wektor Dn

Wektor I zawiera indeksy elementów wektora D w porządku od najmniejszej do największej odległości.

1	2	4	3	5

Tablica 3.6: Wektor I

Nc to wektor zwierający liczby, które określają liczbę klonów dla danego przeciwciała Ab.

20	8	5	1

Tablica 3.7: Wektor Nc

Na przykład przeciwciało o indeksie 1, czyli o największym powinowactwie ma 20 klonów.

krok 3. n wybranych przeciwciał poddawane jest procesowi klonowania proporcjonalnie do ich powinowactwa do antygenów $f_{i,j}$. Wygenerowanie zbioru klonów **C** o najwyższym powinowactwie.

Ta część kodu programu (ang. Clone & Affinity Maturation) określa trzy zbiory:

C zbiór klonów w porządku od największego do najmniejszego stopnia powinowactwa (ang. clones from greater to smaller affinity), Cag zbiór klonów zbioru **Ag** (ang. clones of **Ag**), Cmi zbiór klonów (ang. clones of mi), $k = 0$.

Wywołanie funkcji *clone* w środowisku Matlab:

$$[C, Cag, Cmi] = \\ clone(Ab(:, (k * ma + 1) : (k + 1) * ma), \\ Ag(vet(i), (k * ma + 1) : (k + 1) * ma), mi, D, I, Nc);$$

Funkcja *clone* zwraca zbiory: C,Cag,Cmi. Macierz **C** w funkcji *clone* obliczana jest następująco:

$$C = Ab(I(1), :); \\ Cag = C; \\ vones = ones(Nc(i), 1); \\ C = [C; vones * Ab(I(i), :)];$$

Macierz **C** zawiera: 1 klon przeciwciała Ab_1 ($C = Ab(I(1), :);$), 20 klonów przeciwciała Ab_1, 8 klonów przeciwciała Ab_2, 5 klonów przeciwciała Ab_4, 1 klon przeciwciała Ab_5. Kolejność przeciwciał do klonowania ustala wektor I, a ilość klonów wektor Nc.

L.p													
1	0,00	0,14	0,32	0,54	0,75	0,90	0,99	1,00	0,96	0,90	0,84	0,81	0,82
2	0,00	0,14	0,32	0,54	0,75	0,90	0,99	1,00	0,96	0,90	0,84	0,81	0,82
3	0,00	0,14	0,32	0,54	0,75	0,90	0,99	1,00	0,96	0,90	0,84	0,81	0,82
4	0,00	0,14	0,32	0,54	0,75	0,90	0,99	1,00	0,96	0,90	0,84	0,81	0,82
5	0,00	0,14	0,32	0,54	0,75	0,90	0,99	1,00	0,96	0,90	0,84	0,81	0,82
6	0,00	0,14	0,32	0,54	0,75	0,90	0,99	1,00	0,96	0,90	0,84	0,81	0,82
7	0,00	0,14	0,32	0,54	0,75	0,90	0,99	1,00	0,96	0,90	0,84	0,81	0,82
8	0,00	0,14	0,32	0,54	0,75	0,90	0,99	1,00	0,96	0,90	0,84	0,81	0,82
9	0,00	0,14	0,32	0,54	0,75	0,90	0,99	1,00	0,96	0,90	0,84	0,81	0,82
10	0,00	0,14	0,32	0,54	0,75	0,90	0,99	1,00	0,96	0,90	0,84	0,81	0,82
11	0,00	0,14	0,32	0,54	0,75	0,90	0,99	1,00	0,96	0,90	0,84	0,81	0,82
12	0,00	0,14	0,32	0,54	0,75	0,90	0,99	1,00	0,96	0,90	0,84	0,81	0,82
13	0,00	0,14	0,32	0,54	0,75	0,90	0,99	1,00	0,96	0,90	0,84	0,81	0,82
14	0,00	0,14	0,32	0,54	0,75	0,90	0,99	1,00	0,96	0,90	0,84	0,81	0,82
15	0,00	0,14	0,32	0,54	0,75	0,90	0,99	1,00	0,96	0,90	0,84	0,81	0,82
16	0,00	0,14	0,32	0,54	0,75	0,90	0,99	1,00	0,96	0,90	0,84	0,81	0,82
17	0,00	0,14	0,32	0,54	0,75	0,90	0,99	1,00	0,96	0,90	0,84	0,81	0,82
18	0,00	0,14	0,32	0,54	0,75	0,90	0,99	1,00	0,96	0,90	0,84	0,81	0,82
19	0,00	0,14	0,32	0,54	0,75	0,90	0,99	1,00	0,96	0,90	0,84	0,81	0,82
20	0,00	0,14	0,32	0,54	0,75	0,90	0,99	1,00	0,96	0,90	0,84	0,81	0,82
21	0,00	0,14	0,32	0,54	0,75	0,90	0,99	1,00	0,96	0,90	0,84	0,81	0,82
22	0,00	0,03	0,05	0,07	0,10	0,13	0,19	0,28	0,41	0,58	0,76	0,92	1,00
23	0,00	0,03	0,05	0,07	0,10	0,13	0,19	0,28	0,41	0,58	0,76	0,92	1,00
24	0,00	0,03	0,05	0,07	0,10	0,13	0,19	0,28	0,41	0,58	0,76	0,92	1,00
25	0,00	0,03	0,05	0,07	0,10	0,13	0,19	0,28	0,41	0,58	0,76	0,92	1,00
26	0,00	0,03	0,05	0,07	0,10	0,13	0,19	0,28	0,41	0,58	0,76	0,92	1,00
27	0,00	0,03	0,05	0,07	0,10	0,13	0,19	0,28	0,41	0,58	0,76	0,92	1,00
28	0,00	0,03	0,05	0,07	0,10	0,13	0,19	0,28	0,41	0,58	0,76	0,92	1,00
29	0,00	0,03	0,05	0,07	0,10	0,13	0,19	0,28	0,41	0,58	0,76	0,92	1,00
30	0,32	0,21	0,12	0,05	0,00	0,00	0,05	0,15	0,31	0,50	0,71	0,89	1,00
31	0,32	0,21	0,12	0,05	0,00	0,00	0,05	0,15	0,31	0,50	0,71	0,89	1,00
32	0,32	0,21	0,12	0,05	0,00	0,00	0,05	0,15	0,31	0,50	0,71	0,89	1,00
33	0,32	0,21	0,12	0,05	0,00	0,00	0,05	0,15	0,31	0,50	0,71	0,89	1,00
34	0,32	0,21	0,12	0,05	0,00	0,00	0,05	0,15	0,31	0,50	0,71	0,89	1,00
35	0,57	0,77	0,92	1,00	1,00	0,92	0,74	0,50	0,29	0,15	0,08	0,04	0,00

Tablica 3.8: Macierz **C**

Macierz Cmi w funkcji o identyfikatorze *clone* obliczana jest następująco:

$Cmi = ones(1, L);$
$vones = ones(Nc(i), 1);$
$Cmi = [Cmi; rand(Nc(i), L). * D(I(i)). * mi];$

L.p													
1	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
2	0,59	0,68	0,27	1,09	1,19	1,02	0,61	0,16	0,36	0,97	1,29	0,24	0,95
3	0,20	0,52	0,37	0,65	1,32	0,10	0,33	0,59	0,32	0,51	0,69	0,76	1,12
4	0,83	1,24	0,64	0,98	1,23	0,96	1,11	0,07	0,57	1,20	1,15	0,62	1,04
5	1,01	0,38	0,53	0,95	0,53	0,19	0,40	1,33	0,80	0,09	0,30	0,52	1,33
6	0,37	1,03	0,94	1,11	0,88	0,82	0,04	0,93	0,56	0,35	0,32	0,04	0,78
7	0,70	0,97	0,43	0,01	0,72	0,63	1,32	0,82	1,02	0,83	1,20	0,46	0,48
8	0,42	1,22	0,06	0,06	0,28	0,68	0,64	0,82	0,03	0,91	0,95	1,30	0,50
9	0,24	0,14	1,04	0,67	0,41	0,46	0,80	1,11	0,92	0,98	1,30	0,77	1,32
10	1,04	1,19	0,74	0,39	0,02	0,11	0,69	1,31	1,00	1,24	0,61	1,03	1,18
11	1,08	0,79	0,09	0,02	0,01	0,34	1,30	0,58	1,27	0,82	0,99	0,84	0,25
12	0,96	0,02	0,66	1,06	1,32	0,70	0,33	1,13	0,52	0,98	0,89	0,98	0,33
13	0,06	0,32	1,11	0,09	1,08	0,19	0,89	0,53	0,07	0,85	0,04	0,20	0,33
14	0,82	0,16	0,61	0,41	1,03	0,16	0,20	0,56	0,50	0,23	0,47	0,55	0,23
15	0,45	1,17	1,04	0,45	1,20	0,67	0,54	1,19	0,88	0,36	0,11	0,09	0,45
16	0,78	1,08	0,66	1,09	0,17	0,38	0,19	0,46	0,94	1,08	0,08	1,18	0,54
17	1,22	1,02	0,93	0,01	0,13	0,15	0,46	1,08	0,12	0,54	1,01	1,11	0,83
18	0,67	1,08	1,31	0,66	0,74	0,67	0,73	1,33	1,26	0,88	0,84	0,42	1,20
19	0,56	0,91	0,47	0,82	0,01	0,50	0,06	1,04	0,71	0,38	0,71	1,20	0,91
20	1,07	0,83	0,83	0,50	0,17	0,36	0,31	1,14	1,21	1,28	1,32	1,00	0,18
21	1,19	0,49	0,69	0,26	0,07	1,28	0,53	0,88	0,41	0,77	0,40	0,20	0,63
22	5,81	0,21	1,54	1,24	4,03	0,02	5,10	0,26	4,73	2,04	2,50	2,37	2,78
23	2,99	4,43	4,38	5,25	2,91	5,46	5,79	0,04	1,10	4,12	5,71	0,60	0,80
24	2,38	5,10	0,37	1,42	2,30	3,97	4,88	4,91	3,04	1,31	5,02	2,77	4,57
25	5,93	3,10	0,68	4,11	3,38	0,30	1,59	5,50	0,37	1,39	5,38	3,34	0,81
26	4,61	0,86	4,89	5,77	2,94	2,72	1,24	4,45	0,81	2,66	1,53	1,82	2,59
27	5,90	4,67	2,77	2,56	0,93	0,99	5,60	1,22	0,13	2,00	3,13	3,79	5,16
28	1,44	2,69	5,46	1,05	3,91	2,83	2,36	1,41	5,22	1,60	2,02	3,15	2,80
29	2,80	0,99	5,38	2,39	4,37	4,94	0,37	5,40	3,69	4,53	3,94	1,85	1,72
30	0,47	2,65	5,61	5,18	2,71	3,98	3,73	0,68	0,67	1,04	3,82	1,42	5,29
31	4,71	0,50	0,50	6,36	5,99	4,88	5,74	3,22	5,54	0,67	2,14	3,55	3,72
32	4,52	6,23	4,54	6,73	4,93	3,07	5,59	7,01	2,90	5,04	2,89	2,18	6,45
33	5,25	2,11	0,30	6,64	3,98	1,17	6,66	6,95	2,01	4,89	1,09	1,06	5,07
34	6,85	3,97	0,17	5,43	0,62	4,98	5,21	4,63	1,62	3,28	5,67	6,92	2,83
35	5,75	2,73	0,76	7,93	0,84	8,52	5,54	6,81	5,56	5,19	2,52	0,89	6,46

Tablica 3.9: Macierz Cmi

Macierz Cag w funkcji o identyfikatorze *clone* obliczana jest następująco:

$vones = ones(Nc(i), 1);$
$Cag = [Cag; vones * ag];$

L.p													
1	0,00	0,14	0,32	0,54	0,75	0,90	0,99	1,00	0,96	0,90	0,84	0,81	0,82
2	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
3	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
4	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
5	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
6	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
7	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
8	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
9	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
10	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
11	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
12	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
13	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
14	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
15	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
16	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
17	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
18	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
19	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
20	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
21	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
22	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
23	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
24	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
25	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
26	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
27	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
28	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
29	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
30	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
31	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
32	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
33	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
34	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79
35	0,00	0,15	0,29	0,42	0,56	0,72	0,87	0,97	1,00	0,97	0,90	0,83	0,79

Tablica 3.10: Macierz Cag

krok 4. Elementy zbioru $\mathbf{C}$ poddawane są operatorowi mutacji i w wyniku generowany jest zbiór $\mathbf{C}^*$. Prawdopodobieństwo mutacji jest odwrotnie proporcjonalne do powinowactwa przeciwciał rodzicielskich do j-tego antygenu Ag_j, $f_{i,j}$. Duże powinowactwo implikuje małe prawdopodobieństwo mutacji;

$$\mathbf{C}^* = C - Cmi. * (C - Cag);$$

L.p													
1	0,00	0,14	0,32	0,54	0,75	0,90	0,99	1,00	0,96	0,90	0,84	0,81	0,82
2	0,00	0,15	0,31	0,40	0,53	0,72	0,91	0,99	0,98	0,97	0,92	0,82	0,79
3	0,00	0,15	0,31	0,46	0,50	0,89	0,95	0,98	0,98	0,94	0,89	0,83	0,79
4	0,00	0,15	0,30	0,42	0,52	0,73	0,85	1,00	0,98	0,98	0,91	0,82	0,79
5	0,00	0,15	0,30	0,42	0,65	0,87	0,94	0,96	0,99	0,91	0,86	0,82	0,78
6	0,00	0,15	0,29	0,40	0,58	0,75	0,98	0,97	0,98	0,93	0,86	0,81	0,80
7	0,00	0,15	0,31	0,54	0,62	0,79	0,83	0,97	1,00	0,96	0,92	0,82	0,80
8	0,00	0,15	0,32	0,53	0,70	0,78	0,91	0,97	0,97	0,96	0,90	0,84	0,80
9	0,00	0,15	0,28	0,46	0,67	0,82	0,89	0,96	1,00	0,97	0,92	0,83	0,78
10	0,00	0,15	0,30	0,49	0,75	0,88	0,90	0,96	1,00	0,99	0,88	0,83	0,79
11	0,00	0,15	0,32	0,54	0,75	0,84	0,83	0,98	1,01	0,96	0,90	0,83	0,81
12	0,00	0,14	0,30	0,41	0,50	0,78	0,95	0,96	0,98	0,97	0,90	0,83	0,81
13	0,00	0,15	0,28	0,53	0,55	0,87	0,88	0,98	0,97	0,96	0,85	0,81	0,81
14	0,00	0,15	0,30	0,49	0,56	0,87	0,96	0,98	0,98	0,92	0,87	0,82	0,81
15	0,00	0,15	0,28	0,48	0,52	0,78	0,92	0,96	1,00	0,93	0,85	0,81	0,81
16	0,00	0,15	0,30	0,40	0,72	0,83	0,96	0,99	1,00	0,98	0,85	0,84	0,80
17	0,00	0,15	0,29	0,54	0,73	0,88	0,93	0,97	0,97	0,94	0,91	0,84	0,80
18	0,00	0,15	0,28	0,46	0,61	0,78	0,90	0,96	1,01	0,96	0,89	0,82	0,79
19	0,00	0,15	0,30	0,44	0,75	0,81	0,98	0,97	0,99	0,93	0,89	0,84	0,79
20	0,00	0,15	0,29	0,48	0,72	0,84	0,95	0,96	1,01	0,99	0,92	0,83	0,81
21	0,00	0,15	0,30	0,51	0,74	0,67	0,92	0,97	0,98	0,96	0,87	0,81	0,80
22	0,00	0,06	0,41	0,50	1,98	0,14	3,64	0,46	3,20	1,37	1,12	0,72	0,42
23	0,00	0,57	1,08	1,87	1,46	3,35	4,10	0,31	1,06	2,18	1,59	0,87	0,83
24	0,00	0,66	0,14	0,56	1,17	2,47	3,49	3,66	2,20	1,09	1,49	0,69	0,04
25	0,00	0,41	0,21	1,48	1,68	0,31	1,26	4,06	0,63	1,12	1,54	0,64	0,83
26	0,00	0,14	1,19	2,05	1,47	1,73	1,03	3,34	0,89	1,61	0,98	0,77	0,46
27	0,00	0,60	0,70	0,95	0,53	0,71	3,97	1,12	0,49	1,36	1,22	0,60	-0,08
28	0,00	0,36	1,33	0,43	1,92	1,80	1,79	1,25	3,49	1,20	1,05	0,65	0,41
29	0,00	0,15	1,31	0,89	2,14	3,05	0,44	3,99	2,58	2,34	1,33	0,76	0,64
30	0,17	0,05	1,06	1,96	1,52	2,87	3,09	0,71	0,77	0,99	1,46	0,81	-0,11
31	-1,18	0,18	0,20	2,40	3,36	3,51	4,75	2,78	4,14	0,82	1,13	0,70	0,22
32	-1,12	-0,17	0,88	2,53	2,77	2,21	4,62	5,87	2,31	2,86	1,27	0,77	-0,35
33	-1,35	0,08	0,17	2,50	2,24	0,84	5,49	5,82	1,70	2,78	0,92	0,83	-0,06
34	-1,86	-0,03	0,15	2,05	0,35	3,59	4,31	3,93	1,43	2,04	1,82	0,52	0,41
35	-2,73	-0,91	0,44	-3,63	0,63	-0,75	1,42	3,68	4,26	4,40	2,15	0,75	5,11

Tablica 3.11: Macierz $\mathbf{C}^*$

krok 5. Wyznaczenie powinowactwa $d_{k,j} = 1/D_{k,j}$ pomiędzy Ag_j dla każdego elementu k zbioru $\mathbf{C}^*$:
w tym przykładzie $j = 1$, $k = 0$;

Wywołanie funkcji dist w środowisku Matlab:

$D = dist(\mathbf{C}^*(:, (k * ma + 1) : (k + 1) * ma), Ag(vet(i), (k * ma + 1) : (k + 1) * ma)')$;

gdzie: D wektor odległości pomiędzy $\mathbf{C}^*$ a Ag_j dla $j = 1$;

0,33	0,08	0,21	0,06	0,20
0,14	0,16	0,20	0,16	0,26
0,26	0,12	0,20	0,21	0,13
0,22	0,27	0,10	0,24	0,22
0,22	3,94	4,85	4,50	3,60
3,40	3,42	3,39	4,80	3,84
7,04	7,62	7,65	6,12	8,79

Tablica 3.12: Wektor D

krok 6. Ze zbioru $\mathbf{C}^*$, wybrać $qi[\%]$ przeciwciał, które mają największy wskaźnik powinowactwa $d_{k,j}$ i utworzenie z nich komórek pamięciowych M_j;

Część kodu programu:

$[Dn, I] = sort(D);$
$nR = round(qi * size(C, 1));$
$m = C(I(1 : nR), :);$
$D = D(I(1 : nR));$

Dn	I	Dn	I	Dn	I
0,06	4	0,21	14	3,60	25
0,08	2	0,22	21	3,84	30
0,10	18	0,22	20	3,94	22
0,12	12	0,22	16	4,50	24
0,13	15	0,24	19	4,80	29
0,14	6	0,26	11	4,85	23
0,16	7	0,26	10	6,12	34
0,16	9	0,27	17	7,04	31
0,20	8	0,33	1	7,62	32
0,20	13	3,39	28	7,65	33
0,20	5	3,40	26	8,79	35
0,21	3	3,42	27		

Tablica 3.13: Wektor Dn i wektor I

$nR = 11$

L.p													
1	0,00	0,15	0,30	0,42	0,52	0,73	0,85	1,00	0,98	0,98	0,91	0,82	0,79
2	0,00	0,15	0,31	0,40	0,53	0,72	0,91	0,99	0,98	0,97	0,92	0,82	0,79
3	0,00	0,15	0,28	0,46	0,61	0,78	0,90	0,96	1,01	0,96	0,89	0,82	0,79
4	0,00	0,14	0,30	0,41	0,50	0,78	0,95	0,96	0,98	0,97	0,90	0,83	0,81
5	0,00	0,15	0,28	0,48	0,52	0,78	0,92	0,96	1,00	0,93	0,85	0,81	0,81
6	0,00	0,15	0,29	0,40	0,58	0,75	0,98	0,97	0,98	0,93	0,86	0,81	0,80
7	0,00	0,15	0,31	0,54	0,62	0,79	0,83	0,97	1,00	0,96	0,92	0,82	0,80
8	0,00	0,15	0,28	0,46	0,67	0,82	0,89	0,96	1,00	0,97	0,92	0,83	0,78
9	0,00	0,15	0,32	0,53	0,70	0,78	0,91	0,97	0,97	0,96	0,90	0,84	0,80
10	0,00	0,15	0,28	0,53	0,55	0,87	0,88	0,98	0,97	0,96	0,85	0,81	0,81
11	0,00	0,15	0,30	0,42	0,65	0,87	0,94	0,96	0,99	0,91	0,86	0,82	0,78

Tablica 3.14: Macierz m

0,06	0,08	0,10	0,12	0,13	0,14	0,16	0,16	0,20	0,20	0,20

Tablica 3.15: Wektor D

krok 7. Śmierć komórek: Eliminacja wszystkich komórek pamięciowych ze zbioru M_j dla których wartość powinowactwa spełnia relacje $D_{k,j} > tp$, w tym przykładzie: $tp = 0.5$. Na tym etapie nie zostaną usunięte żadne komórki, ponieważ nie jest spełniona relacja $D_{k,j} > tp$.

Część kodu programu:

```
% Network pruning (Natural Death)
```
$Ip = find(D > tp);$
$m = extract(m, Ip);$
$D = extract(D, Ip);$

Wynik to $Ip = 0$

L.p													
1	0,00	0,15	0,30	0,42	0,52	0,73	0,85	1,00	0,98	0,98	0,91	0,82	0,79
2	0,00	0,15	0,31	0,40	0,53	0,72	0,91	0,99	0,98	0,97	0,92	0,82	0,79
3	0,00	0,15	0,28	0,46	0,61	0,78	0,90	0,96	1,01	0,96	0,89	0,82	0,79
4	0,00	0,14	0,30	0,41	0,50	0,78	0,95	0,96	0,98	0,97	0,90	0,83	0,81
5	0,00	0,15	0,28	0,48	0,52	0,78	0,92	0,96	1,00	0,93	0,85	0,81	0,81
6	0,00	0,15	0,29	0,40	0,58	0,75	0,98	0,97	0,98	0,93	0,86	0,81	0,80
7	0,00	0,15	0,31	0,54	0,62	0,79	0,83	0,97	1,00	0,96	0,92	0,82	0,80
8	0,00	0,15	0,28	0,46	0,67	0,82	0,89	0,96	1,00	0,97	0,92	0,83	0,78
9	0,00	0,15	0,32	0,53	0,70	0,78	0,91	0,97	0,97	0,96	0,90	0,84	0,80
10	0,00	0,15	0,28	0,53	0,55	0,87	0,88	0,98	0,97	0,96	0,85	0,81	0,81
11	0,00	0,15	0,30	0,42	0,65	0,87	0,94	0,96	0,99	0,91	0,86	0,82	0,78

Tablica 3.16: Macierz m

0,06	0,08	0,10	0,12	0,13	0,14	0,16	0,16	0,20	0,20	0,20

Tablica 3.17: Wektor D

krok 8. Wyznaczenie powinowactwa pomiędzy klonowanymi przeciwciałami $s_{i,k} = |Ab_i - Ab_j|$;

krok 9. Supresja: eliminacja tych klonów ze zbioru **M**, dla których spełniona jest relacja $s_{i,k} < ts$, $ts = 0.005$.

Część kodu programu:

```
% Suppression (Idiotypic Network)
% Function suppress self-recognizing
and non-stimulated Ab from Memory M
[m, D1] = suppress(m, ts, pa, ma);
```

Czyli na tym etapie nie zostaną usunięte żadne komórki, ponieważ niespełniona jest relacja $|Ab_i - Ab_j| < ts$

krok 10. Połączenie wszystkich przeciwciał, które są zapisane w zbiorze M_j

Część kodu programu:

```
M = [M; m];
M % pamięć AIS
```

L.p													
1	0,00	0,15	0,30	0,42	0,52	0,73	0,85	1,00	0,98	0,98	0,91	0,82	0,79
2	0,00	0,15	0,31	0,40	0,53	0,72	0,91	0,99	0,98	0,97	0,92	0,82	0,79
3	0,00	0,15	0,28	0,46	0,61	0,78	0,90	0,96	1,01	0,96	0,89	0,82	0,79
4	0,00	0,14	0,30	0,41	0,50	0,78	0,95	0,96	0,98	0,97	0,90	0,83	0,81
5	0,00	0,15	0,28	0,48	0,52	0,78	0,92	0,96	1,00	0,93	0,85	0,81	0,81
6	0,00	0,15	0,29	0,40	0,58	0,75	0,98	0,97	0,98	0,93	0,86	0,81	0,80
7	0,00	0,15	0,31	0,54	0,62	0,79	0,83	0,97	1,00	0,96	0,92	0,82	0,80
8	0,00	0,15	0,28	0,46	0,67	0,82	0,89	0,96	1,00	0,97	0,92	0,83	0,78
9	0,00	0,15	0,32	0,53	0,70	0,78	0,91	0,97	0,97	0,96	0,90	0,84	0,80
10	0,00	0,15	0,28	0,53	0,55	0,87	0,88	0,98	0,97	0,96	0,85	0,81	0,81
11	0,00	0,15	0,30	0,42	0,65	0,87	0,94	0,96	0,99	0,91	0,86	0,82	0,78

Tablica 3.18: Macierz M

3.2.5 Algorytm AIS nr 2

Proponowany algorytm jest uproszczeniem wcześniej opisanego algorytmu AIS nr 1. Uproszczenie to pozwala na wykonanie programu w krótkim czasie w porównaniu do wcześniej prezentowanego algorytmu. Ma to istotne znaczenie w systemach decyzyjnych pracujących w reżimie czasu rzeczywistego. W wielu przypadkach taki system pracy obowiązuje w systemach intensywnej opieki medycznej. Antygen dany jest w postaci wektora. Elementami wektora są wartości wskaźnika pO_2/FiO_2 odpowiednio $Ag(1)$, $Ag(2)$, $Ag(3)$. Wartości tego wektora notowane są w trzech różnych terminach z określonymi indeksami porządku.

Przeciwciała dane są w postaci czterech (w tym przykładzie) wektorów wybranych losowo z dużej grupy wektorów zebranej w bazie danych. Są to:
$Ab_1(1)$, $Ab_1(2)$, $Ab_1(3)$, $Ab_1(4)$
$Ab_2(1)$, $Ab_2(2)$, $Ab_2(3)$, $Ab_2(4)$
$Ab_3(1)$, $Ab_3(2)$, $Ab_3(3)$, $Ab_3(4)$
$Ab_4(1)$, $Ab_4(2)$, $Ab_4(3)$, $Ab_4(4)$

Elementy tych wektorów są wartościami wskaźnika pO_2/FiO_2 w czterech uporządkowanych terminach i zgromadzonych w bazie danych. Zadaniem jest znaleźć brakującą wartość parametru w postaci elementu $Ag(4)$ w przyszłym terminie dla wybranego wskaźnika. Parametr ten opisuje nieznaną wartość w przyszłej chwili czasu określoną indeksem (4).

Krok1. Obliczamy odległości według wzoru. Są to cztery liczby $d(i)$
$dla\, i = 1, 2, 3, 4$, których odległości oblicza się według wzorów:

$$d(1) = 1/(3*4)*($$

$$|Ag(1)-Ab_1(1)|+|Ag(1)-Ab_1(2)|+|Ag(1)-Ab_1(3)|+|Ag(1)-Ab_1(4)|+$$
$$|Ag(2)-Ab_1(1)|+|Ag(2)-Ab_1(2)|+|Ag(2)-Ab_1(3)|+|Ag(2)-Ab_1(4)|+$$
$$|Ag(3)-Ab_1(1)|+|Ag(3)-Ab_1(2)|+|Ag(3)-Ab_1(3)|+|Ag(3)-Ab_1(4)|)$$

$$d(2) = 1/(3*4)*($$

$$|Ag(1)-Ab_2(1)|+|Ag(1)-Ab_2(2)|+|Ag(1)-Ab_2(3)|+|Ag(1)-Ab_2(4)|+$$
$$|Ag(2)-Ab_2(1)|+|Ag(2)-Ab_2(2)|+|Ag(2)-Ab_2(3)|+|Ag(2)-Ab_2(4)|+$$
$$|Ag(3)-Ab_2(1)|+|Ag(3)-Ab_2(2)|+|Ag(3)-Ab_2(3)|+|Ag(3)-Ab_2(4)|)$$

$$d(3) = 1/(3*4)*($$

$$|Ag(1)-Ab_3(1)|+|Ag(1)-Ab_3(2)|+|Ag(1)-Ab_3(3)|+|Ag(1)-Ab_3(4)|+$$
$$|Ag(2)-Ab_3(1)|+|Ag(2)-Ab_3(2)|+|Ag(2)-Ab_3(3)|+|Ag(2)-Ab_3(4)|+$$
$$|Ag(3)-Ab_3(1)|+|Ag(3)-Ab_3(2)|+|Ag(3)-Ab_3(3)|+|Ag(3)-Ab_3(4)|)$$

$$d(4) = 1/(3*4)*($$

$$|Ag(1)-Ab_4(1)|+|Ag(1)-Ab_4(2)|+|Ag(1)-Ab_4(3)|+|Ag(1)-Ab_4(4)|+$$

$|Ag(2) - Ab_4(1)| + |Ag(2) - Ab_4(2)| + |Ag(2) - Ab_4(3)| + |Ag(2) - Ab_4(4)|+$
$|Ag(3) - Ab_4(1)| + |Ag(3) - Ab_4(2)| + |Ag(3) - Ab_4(3)| + |Ag(3) - Ab_4(4)|)$

Wartość $d(i)$ wynika z odejmowania każdego Ag od każdego Ab. Mamy dwanaście wartości różnic bezwzględnych. Problemem jest to, że wektory nie są równej długości.

Krok 2. Obliczmy najmniejszą odległość ze zbioru $d(i)$, $i = 1, 2, 3, 4$ wybieramy ten wektor danych Ab, który spełnia $min(d(1), d(2), d(3), d(4))$ ten wektor zapamiętamy w zbiorze $\mathbf{C}^*$.

Krok 3. Wybieramy losowo inną grupę wektorów z całej populacji powtarzamy całą procedurę w taki sam sposób aby wybrać drugi wektor do zbioru $\mathbf{C}^*$.

Krok 4. Procedurę realizujemy n razy i w zbiorze $\mathbf{C}^*$ otrzymamy n wektorów. Dla tych n wektorów mamy n wartości $Ab_i(4)$ dla $i = 1, 2, ...n$.

Krok 5. Wybieramy dwie wartości: $min(Ab_i(4))$ dla $i = 1, 2, ...n$ oraz $max(Ab_i(4))$ dla $i = 1, 2, ...n$

Krok 6. Określamy przedział mutacji (mutacja jest realizowana metodą ruletki, poprzez losowanie, podobnie jak w algorytmie genetycznym) dla brakującego elementu antygenu $Ag(4)$ według wzoru:

$Ag(4) = random$(od wartości $min(Ab_i(4))$ dla $i = 1, 2, ...n$ do wartości $max(Ab_i(4))$ dla $i = 1, 2, ...n$)

Wybrany element $Ag(4)$ losujemy m razy i otrzymujemy m zmutowanych wartości dla elementów brakujących dla nieznanej wartości Ag w terminie oznaczonym indeksem (4). Otrzymamy wektor, w którym trzy pierwsze w kolejności elementy są znane, a czwarty jest określoną procedurą:

$Ag(1)$, $Ag(2)$, $Ag(3)$, $random$(od wartości $min(Ab_i(4))$ dla $i = 1, 2, ...n$ do wartości $max(Ab_i(4))$ dla $i = 1, 2, ...n$)

$Ag(1)$, $Ag(2)$, $Ag(3)$, $random$(od wartości $min(Ab_i(4))$ dla $i = 1, 2, ...n$ do wartości $max(Ab_i(4))$ dla $i = 1, 2, ...n$)

$Ag(1)$, $Ag(2)$, $Ag(3)$, $random$(od wartości $min(Ab_i(4))$ dla $i = 1, 2, ...n$ do wartości $max(Ab_i(4))$ dla $i = 1, 2, ...n$)

$Ag(1)$, $Ag(2)$, $Ag(3)$, $random$(od wartości $min(Ab_i(4))$ dla $i = 1, 2, ...n$ do wartości $max(Ab_i(4))$ dla $i = 1, 2, ...n$)

....$Ag(1)$, $Ag(2)$, $Ag(3)$, $random$(od wartości $min(Ab_i(4))$ dla $i = 1, 2, ...n$ do wartości $max(Ab_i(4))$ dla $i = 1, 2, ...n$)

Takich wektorów jest m.

Krok 7. Pamiętamy te wektory w zbiorze dedykowanym dla parametru pO_2/FiO_2.

Krok 8. Powtarzamy całą procedurę dla innego niż pO_2/FiO_2 parametru. Na przykład dla parametru pH, albo HCO_3. W rezultacie otrzymamy m nowych wektorów dla tego nowego parametru, na przykład pH.

Krok 9. Łączymy te wektory w pary, tych par będzie $m * m$, czyli m^2, każdy wektor zawiera 4 elementy.

Krok 10. Definiujemy miarę odległości D według na przykład miary euklidesowej i obliczamy odległości tej pary wektorów od wszystkich takich par, które są w całej bazie danych i odpowiadają tym samym przedziałom czasu obserwacji danych oraz w określonych godzinach obserwacji.

Krok 11. Znajdujemy najmniejszą odległość ze wszystkich D dla całej bazy danych.

Krok 12. Dla znalezionej w kroku 11 najmniejszej odległości przyjmujemy jako wartości przewidywane czyli: $Ag(4)$. Te wartości przyjmujemy jako wartości predykowane dla wybranego parametru.

Obliczenia można powtórzyć dla innych dwóch parametrów, albo dodać jeszcze inny parametr na przykład pO_2.

3.3 Predykcja wskaźnika pO_2/FiO_2 przy użyciu sztucznej sieci immunologicznej z zastosowaniem algorytmu AIS nr 1

3.3.1 Struktura zbioru uczącego

Zbiór uczący dla algorytmu sieci immunologicznej AIS zawiera wartości ułożone w postaci szeregu terminów, w których zapisano wartości parametrów krwi pO_2/FiO_2, pH, pCO_2, HCO_3. Szeregi te są dane w postaci wektorów o takich samych długościach. Każdy wektor uczący składa się z elementów składowych odpowiadających poszczególnym parametrom.

Nr Ag	wartości 1 parametru	wartości 2 parametru	wartości 3 parametru	wartości 4 parametru
1	$pO_2/FiO_2(t_1, t_2...t_{29})$	$pH(t_1, t_2...t_{29})$	$pCO_2(t_1, t_2...t_{29})$	$HCO_3(t_1, t_2...t_{29})$
2	$pO_2/FiO_2(t_1, t_2...t_{29})$	$pH(t_1, t_2...t_{29})$	$pCO_2(t_1, t_2...t_{29})$	$HCO_3(t_1, t_2...t_{29})$
...	...	...	...	...
164	$pO_2/FiO_2(t_1, t_2...t_{29})$	$pH(t_1, t_2...t_{29})$	$pCO_2(t_1, t_2...t_{29})$	$HCO_3(t_1, t_2...t_{29})$

Tablica 3.19: Zbiór do nauki AIS

Każdy szereg terminów $(t_1, t_2...t_{29})$ zawarty w zbiorze uczącym można interpretować jak współrzędne punktu w przestrzeni o wymiarze zależnym od długości wzorca. Taka interpretacja elementów szeregu terminów, pozwala zastosować do obliczania powinowactwa relacji antygen **Ag** - przeciwciało **Ab** według wzorów 3.3 3.2 3.4 3.5. W omawianym przykładzie powinowactwo było obliczane zgodnie ze wzorem 3.2 lub 3.5. W algorytmie uczenia sztucznej sieci immunologicznej [2] został zmieniony sposób obliczania powinowactwa. Powinowactwo zostało obliczone jak suma powinowactw względem każdego parametru wektora uczącego. Wzór obliczania odległości:

$$D = \sum_{i=1}^{i=p} D_i \tag{3.6}$$

gdzie: p liczba parametrów wektora uczącego

Elementy zbioru uczącego są przez sieć interpretowane jako antygeny, które będą stymulowały odpowiednie przeciwciało.

3.3.2 Segmentacja wektora uczącego

Do przeprowadzenia procesu predykcji przy wykorzystaniu sztucznej sieci immunologicznej wektory danych uczących należy poddać segmentacji [30, 31]. Segmentacja polega na podziale całych wektorów uczących zadanym oknem czasowym *(window)*. W tablicach został przedstawiony przykład procesu segmentacji dla wektora uczącego składającego się z dwóch parametrów. Każdy parametr zawiera pięć pomiarów. Szerokość okna czasowego wynosi dwa pomiary. W tablicy 3.20 przedstawiono wektor uczący We przed procesem segmentacji . Wektor zawiera dwa parametry. Po procesie segmentacji dla jednego wektora uczącego We uzyskujemy cztery wektory uczące Wen o liczbie pomiarów równej szerokości okna czasowego.

$Parametr_1$					$Parametr_2$				
x_1	x_2	x_3	x_4	x_5	y_1	y_2	y_3	y_4	y_5

Tablica 3.20: Wektor We przed segmentacją

Pierwszy, drugi, trzeci i czwarty wektor uzyskany w procesie segmentacji zamieszczono w tablicy 3.21.

$Parametr_1$		$Parametr_2$	
x_1	x_2	y_1	y_2
x_2	x_3	y_2	y_3
x_3	x_4	y_3	y_4
x_4	x_5	y_4	y_5

Tablica 3.21: Pierwszy, drugi, trzeci i czwarty wektor Wen

Algorytm segmentacji danych dla sieci immunologicznej

Dla każdego wektora uczącego We w postaci szeregu czasowego, który należy do zbioru uczącego Au przeprowadzamy operacje, które są opisane w przedstawionym poniżej algorytmie segmentacji:

$\forall We_i$ $i = 1, 2, ..., l_w$ wykonaj: krok1 i krok2.

krok 1. dla $j = 1, ..., m - window + 1$ wykonaj przemieszczenie okna czasowego z krokiem 1 do wartości $m - window + 1$ dla p parametrów wektora uczącego;

krok 2. Utworzenie nowego wektora uczącego $Wen_{j,k}$ dla $k = p * window$ zawierającego p okien dla ustalonego j ;

krok 3. Utworzenie nowego zbioru uczącego $Aun_{n,k}$ dla $n = l_w * m - window + 1$, $k = p * window$ zawierający wszystkie wektory Wen**,**

gdzie:

Au zbiór uczący przed procesem segmentacji,

Aun zbiór uczący po procesie segmentacji,

We wektor uczący $We \in Au$ przed procesem segmentacji,

Wen wektor uczący po procesie segmentacji utworzony z p okien otrzymanych w procesie segmentacji,

l_w liczba wektorów uczących przed procesem segmentacji,

l_k liczba pomiarów zapisana w wektorze uczącym przed procesem segmentacji $l_k = p * m$,

gdzie m to liczba pomiarów w dostępnych terminach dla wybranego parametru zawarta w zbiorze uczącym Au,

p liczba parametrów w wektorze uczącym We na przykład dla czterech parametrów opisanych jako: (pO_2/FiO_2, pH, pCO_2, $HCO_{3)}$,

$window$ szerokość okna czasowego.

3.3.3 Proces testowania sieci

Sieć immunologiczna [2] umożliwia klasyfikację szeregów czasowych. Aby była możliwa predykcja przez zastosowanie sztucznej sieci immunologicznej został zmieniony algorytm odpowiedzi sieci [30, 31]. Proces predykcji jest związany ze sprawdzeniem zdolności do predykcji sieci poprzez podanie na wejście sieci części przebiegu ze zbioru testującego. W ten sposób jest obserwowana odpowiedź sieci. Zbiór testujący zawiera przebiegi, które nie należą do zbioru uczącego.

Algorytm predykcji sieci

Zbiorem testującym jest zbiór antygenów $Agt_{i,j}$

$dla\, i = 1, ..., n,\ j = 1, ..., window$.

Dla każdego antygenu $Agt_{i,j}$ dla $i = 1, ..., n,\ j = 1, ..., subwindow\, input$, który należy do zbioru testującego Agt przeprowadzamy operacje, które są opisane w kroku 1 w przedstawionym poniżej algorytmie odpowiedzi sieci. Efektem działania algorytmu jest: dla i-tego antygenu ze zbioru testującego $Agt_{i,j}\, dla\, i = 1, ..., n,\ j = 1, ..., subwindow\, input$ wyznaczana jest odpowiedź sieci. Odpowiedzią jest przeciwciało $Ab_{i,j}$ dla $i = 1, ..., n$ oraz $j = window$, które charakteryzuje się największym powinowactwem.

Algorytm można opisać następująco:

$\forall Agt_{i,j}$ dla $i = 1, ..., n$ oraz $j = 1, ..., subwindow\, input$ wykonaj:

krok 1. dla $k = subwindow\, input, ..., window - 1$ wykonaj:

Wyznacz odpowiedź sieci. Odpowiedzią jest znalezione przeciwciało $Ab_{i,j}$

$dla\, j = k + 1$ o największym powinowactwie.

- W rezultacie obliczeń otrzymano przeciwciało $Ab_{i,j}$

$dla\, i = 1, ..., n$ oraz $j = window$

gdzie:

Agt zbiór antygenów, zbiór wektorów testujących,

n liczba antygenów,

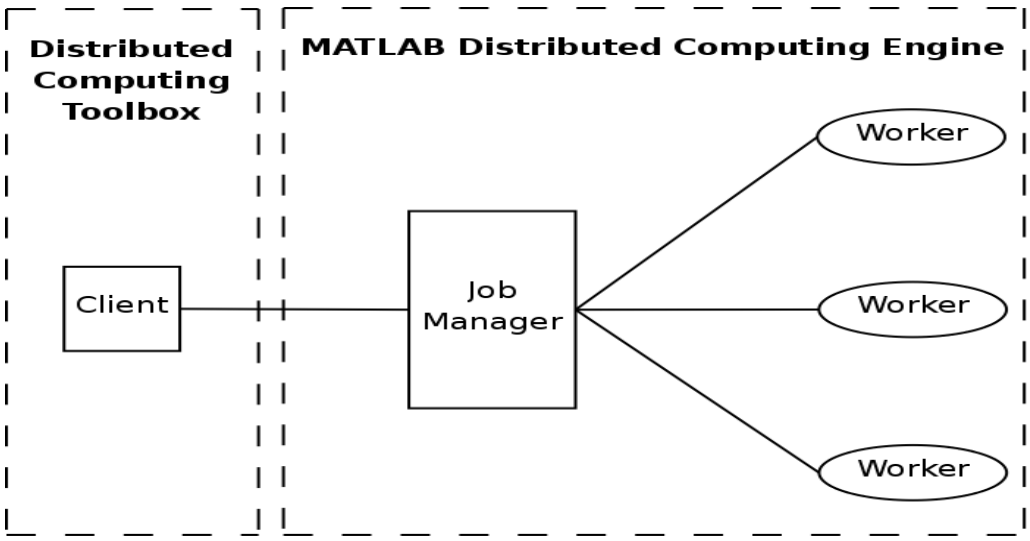
Ab odpowiedź sieci, przeciwciało w postaci wektora, którego parametry stanowią o największym powinowactwie według przyjętej arbitralnie miary obliczania odległości,

window szerokość okna czasowego,

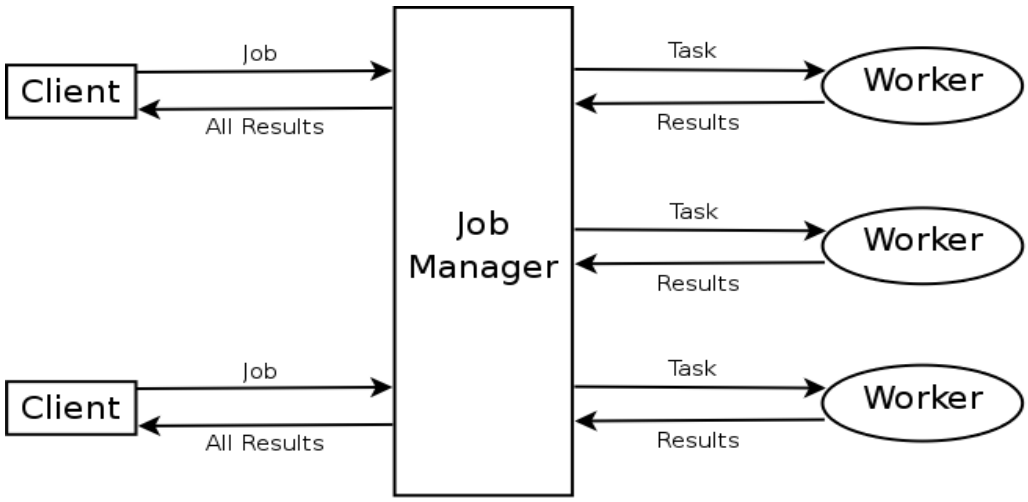
subwindow input podokno czasowe danych wejściowych.

3.4 Obliczenia w środowisku typu klaster MDC

W procesie uczenia sztucznej sieci immunologicznej jest realizowane obliczanie odległości związane z miarą powinowactwa pomiędzy antygenami i przeciwciałami. Proces uczenia sztucznej sieci immunologicznej został przeprowadzony w środowisku rozproszonym o nazwie *klaster MATLAB Distributed Computing (MDC)*. W skład tego środowiska wchodzą pakiety *MATLAB Distributed Computing Engine* oraz *MATLAB Distributed Computing Toolbox*. Pierwszy z nich (MDCE) odpowiedzialny jest za zarządzanie zadaniami w uruchamianych programach wielozadaniowych oraz za zapewnienie komunikacji między nimi w środowisku rozproszonym. Drugi (MDCT), to zestaw funkcji i poleceń umożliwiający użytkownikowi korzystanie z MDCE. Rysunek 3.3



Rysunek 3.3: Środowisko "Basic Distributed Computing Configuration"



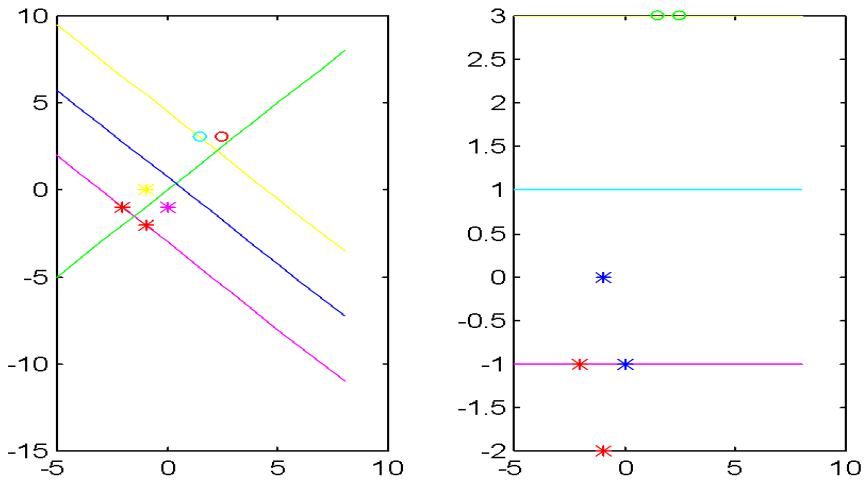
Rysunek 3.4: Środowisko "Interactions of Distributed Computing Sessions"

Licencja ewaluacyjna obejmowała korzystanie z jednego pakietu MDCT oraz z pakietu MDCE na ośmiu procesorach, osiem licencji dla maszyn typu worker. Praca (ang. job) jest operacją, którą użytkownik wykonuje podczas działania sesji Matlab. Praca dzieli się na segmenty zwane zadaniami (ang. tasks). Zadanie jest to najmniejsza porcja operacji, wykonywana na danym komputerze. Zadania są rozpraszane na poszczególne jednostki obliczeniowe zwane workerami. Są to komputery realizujące zadania. Klient to jednostka komputerowa, z której użytkownik żąda realizacji obliczeń. JobManager jest to część pojęcia nazwanego "rozproszoną maszyną obliczeniową środowiska Matlab". JobManager zarządza pracami. Głównym zadaniem JobManagera jest rozproszenie zadań na poszczególne komputery. Komputer liczący (ang. worker) to komputer, w którym uruchamiana jest sesja Matlaba. Głównym zadaniem workerów, jest przyjmowanie i wykonywanie zadań przydzielanych przez JobManagera. Wyniki obliczeń są przekazywane do JobManagera, który przekazuje wyniki do klienta. Zadania rozproszone nie są od siebie zależne i nie ma między nimi komunikacji. Każdy worker pracuje w przybliżeniu tak samo jak zwykła sesja MATLAB'a. Należy pamiętać o niewielkiej stracie czasu związanej z samym uruchomieniem i zatrzymaniem workera na przydzielonym węźle klastra. Konfiguracja rozproszona, ze względu na swoje przeznaczenie, spełnia się najlepiej w przypadku kilku dużych zadań obliczeniowych, wymagających długiego czasu pracy. Współpraca między klientem, a JobManagerem oraz workerem jest przedstawiona na rysunku 3.4. Plan pracy *Matlab Distributed Computing* rozpoczyna się gdy użytkownik uruchamia sesję Matlaba na komputerze klienta. Za pomocą polecenia użytkownik wyszukuje JobManagera wraz z dostępnymi workerami, które są przypisane do JobManagera. Użytkownik tworzy pracę, którą rejestruje Job Manager. Po utworzeniu pracy użytkownik tworzy zadania w nowo utworzonej pracy. JobManager zleca wykonanie zadań workerom. Rezultaty pracy workerów są zwracane do JobManagera a następnie do klienta.

Rozdział 4

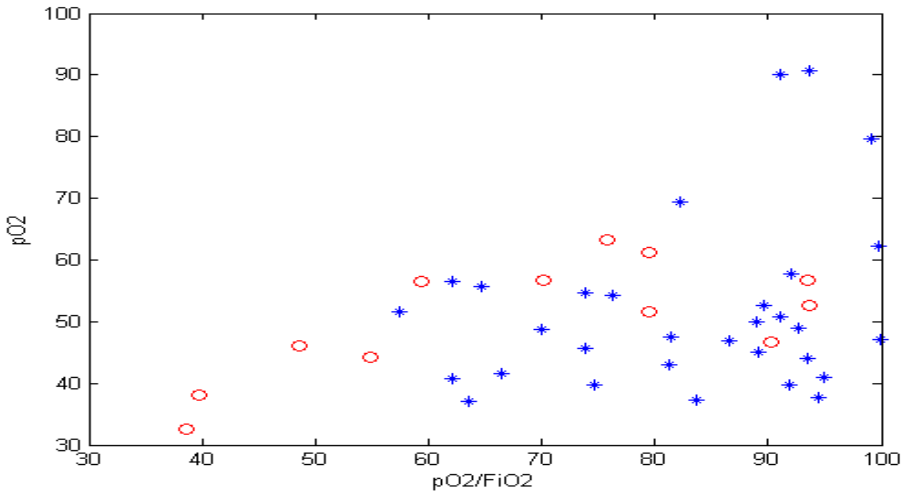
Metoda wektorów nośnych - Support Vector Machines SVM

Metoda SVM została zastosowana do oceny ryzyka zgonu pacjenta. Zbiór danych można podzielić na dwa podzbiory. Elementy należące do pierwszego podzbioru otrzymają etykietę 1, a elementy należące do drugiego podzbioru otrzymają etykietę -1. U podstaw metody wektorów nośnych (Support Vector Machines - SVM) [20, 13] leży koncepcja przestrzeni decyzyjnej, którą dzieli się budując granice separujące obiekty o danej na etapie uczenia i oczekiwanej na etapie testowania przynależności do określonej klasy. Przykład podziału przedstawiono na rysunku 4.1. Mamy dwie klasy graficznie obrazowane przez znak kółka i znak gwiazdki. Linia graniczna rozdziela je wyraźnie. Nowy, nieznany obiekt, jeżeli znajdzie się po prawej stronie granicy zostanie zaklasyfikowany jako kółko, a w przeciwnym wypadku jako gwiazdka.

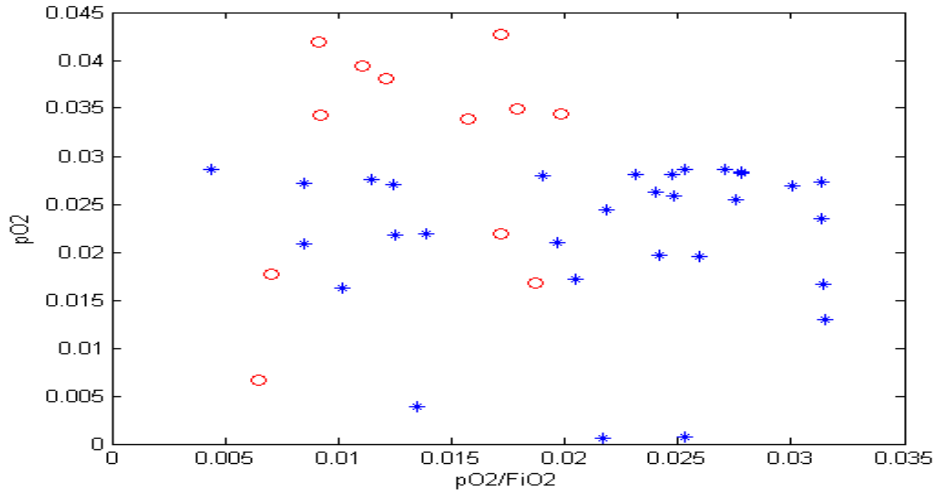


Rysunek 4.1: Odpowiedni podział z maksymalnym marginesem

Rysunek jest ilustracją bardzo prostego przykładu klasyfikatora liniowego, dzielącego obszar klasyfikowany na dwie części za pomocą prostej. Większość praktycznych zadań klasyfikacyjnych jednak nie jest tak łatwa do rozdzielenia za pomocą prostej. Do poprawnego klasyfikowania potrzebne są bardziej skomplikowane formy niż linia prosta. Przykładem może być rysunek 4.2, który porównany z poprzednim jasno wskazuje, że do rozdzielenia różnych znaków potrzebna jest krzywa separująca. Krzywa, podobnie jak prosta są przykładami klasyfikatorów hiperpłaszczyznowych w przestrzeniach wielowymiarowych. Tego typu klasyfikatory otrzymujemy stosując metodę SVM. Rysunki 4.2 i 4.3 ilustrują główną ideę metody SVM. Oryginalne parametry zamieszczone na rysunku 4.2 zostały przetransformowane za pomocą funkcji jądrowej w przestrzeń przedstawioną na rysunku 4.3. W nowej przestrzeni dwie klasy są separowalne.



Rysunek 4.2: Oryginalna przestrzeń parametrów



Rysunek 4.3: Zastosowanie funkcji jądrowej Φ

Metoda wektorów nośnych SVM realizuje zadania klasyfikacyjne konstruując w wielowymiarowej przestrzeni hiperpłaszczyzny oddzielające przypadki należące do różnych klas. Możemy również wykonać regresję, dla wielu zmiennych ciągłych i skategoryzowanych. Dla każdej zmiennej skategoryzowanej tworzony jest zestaw zmiennych z kodami określającymi przynależność każdego przypadku albo -1 albo 1. Na przykład zmienna przyjmująca trzy wartości: A, B i C reprezentowana będzie przez trzy zmienne o wartościach, odpowiednio:

$A : 100$, $B : 010$, $C : 001$.

Optymalną hiperpłaszczyznę separującą buduje się w iteracyjnym algorytmie uczącym, minimalizującym zadaną funkcję błędu.

Modele SVM dzieli się według typu funkcji błędu. Wyróżniamy cztery grupy modeli:

Klasyfikacyjny typ 1 (klasyfikacja C-SVC)
Klasyfikacyjny typ 2 (klasyfikacja ν -SVC)
Regresyjny typ 1 (regresja ϵ-SVR)
Regresyjny typ 2 (regresja ν -SVR)
Poniżej, krótkie omówienie wyszczególnionych typów.

4.1 Klasyfikacja SVM typu 1

Przy tym typie klasyfikacji, w trakcie uczenia modelu minimalizowana jest funkcja błędu w postaci

$$\frac{1}{2}\mathbf{w}^T \cdot \mathbf{w} + C\sum_{i=1}^{i=N}\xi_i \tag{4.1}$$

podlegająca ograniczeniom

$$y_i(\mathbf{w}^T\Phi(x_i) + b) \geq 1 - \xi_i \text{ gdy } \xi_i \geq 0, \text{ dla } i = 1, 2, ..., N \tag{4.2}$$

gdzie C jest stałą, która jest nazywana pojemnością, $\mathbf{w}$ jest wektorem współczynników, zwany też wektorem wag, b jest stałą, a ξ_i parametr adresowany do poszczególnego przypadku. Indeks i numeruje N przypadków zastosowanych w procesie uczenia SVM. $y_i \in (-1; 1)$ są etykietami klas, a x_i to zmienne niezależne. Funkcja jądrowa Φ przekształca dane wejściowe do nowej przestrzeni cech. C ma wpływ na błąd i wartość tego parametru dobierana musi być precyzyjnie, ze względu na problem nadmiernego dopasowania modelu.

4.2 Klasyfikacja SVM typu 2

W tym przypadku, minimalizowana jest funkcja błędu postaci, gdy ν oraz ρ są parametrami metody:

$$\frac{1}{2}\mathbf{w}^T \cdot \mathbf{w} - \nu \cdot \rho + \frac{1}{N}\sum_{i=1}^{i=N} \xi_i \tag{4.3}$$

podlegająca ograniczeniom

$$y_i(\mathbf{w}^T \Phi(x_i) + b) \geq \rho - \xi_i,\ \xi_i \geq 0, i = 1, 2, ..., N, \rho \geq 0. \tag{4.4}$$

4.3 Regresja SVM

W regresji SVM poszukujemy zależności funkcyjnej zmiennej zależnej y od zbioru zmiennych niezależnych x. Przyjmuje się, że zależność ta jest typu deterministycznego f, z dodatkiem losowego szumu w postaci

$$y = f(\mathbf{x}) + szum \tag{4.5}$$

Podstawowym zadaniem jest, więc znalezienie postaci funkcji $f()$, która powinna możliwie najlepiej określić wartość zmiennej zależnej dla nieznanych wcześniej przypadków. Zadanie to rozwiązuje się ucząc model SVM za pomocą próby przypadków, zwanej próbą uczącą metody SVM. Podobnie jak przy klasyfikacji, proces ten polega na sekwencyjnym minimalizowaniu danej funkcji błędu.

Prosty algorytm uczenia napisany w pseudokodzie w celu znalezienia funkcji $f()$ zapisano poniżej.

```
argumenty dla procesu uczenia:
```

$$\mathbf{X} = \{x_1, x_2, ..., x_m\} \subset \chi$$
$$Y = \{y_1, y_2, ..., y_m\} \subset \{+1, -1\}$$
$$Learning - rate = \eta$$

```
wektor wag oraz próg klasyfikacji zwracany przez
algorytm
```

$$\mathbf{w}, b$$
$$function\, learning - perceptron(\mathbf{X}, \mathbf{Y}, \eta)$$
$$initialize\ w, b = 0$$
$$repeat$$
$$for\, \forall i \in \{1, 2, ..., m\}$$
$$compute\ \ g(x_i) = \mathrm{sgn}((\mathbf{w} \cdot x_i) + b)$$
$$update\ \ w := w + (\eta/2)(y_i - g(x_i))x_i$$
$$update\ \ b := b + (\eta/2)(y_i - g(x_i))$$
$$endfor$$
$$until\ \forall\ 1 \leq i \leq m\ we\ have\, g(x_i) = y_i$$
$$return\ \ f : \mathbf{x} \mapsto (\mathbf{w} \cdot \mathbf{x}) + b$$
$$end$$

Stosuje się dwa typy tej funkcji. Odpowiednio, są dwa typy regresji SVM:

4.4 Regresja SVM typu 1

Regresja SVM typu 1 minimalizuje funkcję błędu następującej postaci

$$\frac{1}{2}\mathbf{w}^T \cdot \mathbf{w} + C\sum_{i=1}^{i=N}\xi_i + C\sum_{i=1}^{i=N}\xi_i^* \tag{4.6}$$

przy spełnieniu warunków

$$-y_i + \mathbf{w}^T\Phi(x_i) + b \leq \epsilon + \xi_i^*,$$
$$y_i - \mathbf{w}^T\Phi(x_i) - b \leq \epsilon + \xi_i,\ \ \xi_i, \xi_i^* \geq 0, i = 1, 2, ..., N. \tag{4.7}$$

4.5 Regresja SVM typu 2

Regresja SVM typu 2 określa funkcję błędu w postaci

$$\frac{1}{2}\mathbf{w}^T \cdot \mathbf{w} - C(\nu\epsilon + \frac{1}{N}\sum_{i=1}^{i=N}(\xi_i + \xi_i^*) \tag{4.8}$$

przy spełnieniu warunków

$$(\mathbf{w}^T\Phi(x_i) + b) - y_i \leq \epsilon + \xi_i,$$
$$y_i - (\mathbf{w}^T\Phi(x_i) + b) \leq \epsilon + \xi_i^*,\ \ \xi_i, \xi_i^* \geq 0, i = 1, 2, ..., N. \tag{4.9}$$

4.6 Funkcje jądrowe Φ

Jest wiele typów funkcji jądrowych do zastosowania w modelu SVM: liniowa, wielomianowa, radialne funkcje bazowe (RBF) i funkcja sigmoidalna.

Funkcja liniowa ma postać

$$\Phi = \Phi(x, x). \tag{4.10}$$

Funkcja wielomianowa:

$$\Phi = \Phi(x, \gamma \cdot x \cdot x + \text{const}). \tag{4.11}$$

Funkcja radialna:

$$\Phi = \Phi(x_i, x_j, \exp(-\gamma)|x_i - x_j|^2). \tag{4.12}$$

Funkcja sigmoidalna:

$$\Phi = \Phi(x, \frac{1}{1 + \exp(-x)}). \tag{4.13}$$

RBF jest najczęściej wybieraną jako funkcja jądrowa dla modelu SVM. W pracy wykorzystano model $C - SVC$ (C-suport vector classification) o funkcji jądra RBF w postaci ogólnej

$$\Phi(u, v, \exp(-\gamma \cdot |u - v|^2)). \tag{4.14}$$

Parametr C i parametr funkcji jądrowej γ zostały dobrane automatycznie przy użyciu metody walidacji danych.

Przegląd zadań Support Vector Machines pozwala znaleźć najlepsze zastosowanie metody i wskazać jej zalety w określonych zadaniach oraz wyjaśnić przydatność w semantycznej klasyfikacji. Metoda SVM jest tematem pracy w badaniach przeżywalności dzieci.

4.7 Sformułowanie problemu klasyfikacji

Zakłada się, że dany jest zbiór danych dla procesu uczenia w postaci

$$X := \{x_1, x_2, ..., x_m\} \subseteq R^N \tag{4.15}$$

łącznie z przypisanymi do elementów tego zbioru etykietami

$$Y := \{y_1, y_2, ..., y_m\} \subseteq \{-1, +1\}. \tag{4.16}$$

Celem metody jest znalezienie funkcji decyzyjnej

$$g : R^N \to \{-1, +1\} \tag{4.17}$$

takiej, że jej użycie dokładnie określa etykiety dla nieznanych punktów x w postaci wyniku oznaczonego symbolem y. Poszukiwana jest taka funkcja $g()$, która minimalizuje prawdopodobieństwo wystąpienia

$$g(\mathbf{x}) \neq y,$$

czyli błędnego wyniku klasyfikacji. Do określenia funkcji $g()$ dochodzi się poprzez określenie funkcji

$$f() : R^N \rightarrow R,$$

której zwracana wartość poddawana jest operacji porównania z zadaną wartością progu w celu otrzymania finalnej klasyfikacji

$$g(\mathbf{x}) = \text{sgn}(f(\mathbf{x})).$$

W prostej postaci funkcja $f()$ może być określona następująco

$$g(\mathbf{x}) = \text{sgn}(f(\mathbf{x})) \; gdzie: \; f(\mathbf{x} \cdot \mathbf{w} + b) \; dla \; \mathbf{w} \in R^N \text{ oraz } b \in R. \tag{4.18}$$

4.8 Opis matematyczny

Należy wyznaczyć optymalną hiperpłaszczyznę dzielącą zbiór S w taki sposób aby:

- punkty jednej klasy znalazły się po tej samej stronie hiperpłaszczyzny
- wyznaczyć maksymalny margines czyli dystans pomiędzy punktami danej klasy, a hiperpłaszczyzną rozdzielającą.

Rozważmy punkty zbioru S:

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n) \tag{4.19}$$

gdzie y_i ma wartość 1 lub -1, oraz określa przynależność x_i do danej klasy i jest wektorem, zazwyczaj znormalizowanym. Zbiór S jest liniowo separowalny jeśli istnieje $\mathbf{w} \in R^N$ oraz $b \in R$ takie, że:

$$c_i(\mathbf{w} \cdot \mathbf{x} + b) \geq 1 \tag{4.20}$$

dla

$$i = 1, 2, \ldots, N.$$

Para $(\mathbf{w}, b)$ definiuje hiperpłaszczyznę o równaniu:

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \tag{4.21}$$

Punkty klasyfikowane są do odpowiedniej klasy na podstawie nierówności:

$$\mathbf{w} \cdot x_i + b \geq 1 \tag{4.22}$$

$$\mathbf{w} \cdot x_j + b \leq -1 \tag{4.23}$$

lub ogólnie:

$$c_i(\mathbf{w} \cdot x_i + b) \geq 1, \qquad 1 \leq i \leq N. \tag{4.24}$$

Z prostych zależności geometrycznych wyprowadzamy

$$\mathbf{w} \cdot x_1 + b = 1 \tag{4.25}$$

$$\mathbf{w} \cdot x_2 + b = -1 \tag{4.26}$$

otrzymamy

$$\mathbf{w} \cdot (x_1 - x_2) = 2 \tag{4.27}$$

przez podzielenie obu stron równania otrzymujemy

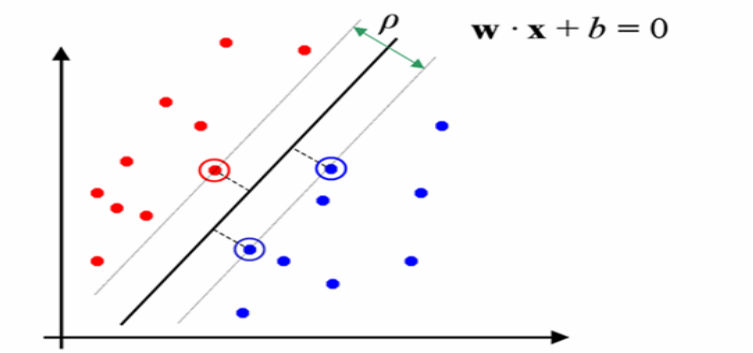
$$\left(\frac{\mathbf{w}}{||\mathbf{w}||} \cdot (x_1 - x_2) \right) = \frac{2}{||\mathbf{w}||} \tag{4.28}$$

odległość punktu x_i od hiperpłaszczyzny jest równa

$$d(x_i) = \frac{\mathbf{w} \cdot x_i + b}{||\mathbf{w}||} \tag{4.29}$$

a szerokość marginesu separującego wynosi:

$$M = \frac{2}{||\mathbf{w}||}. \tag{4.30}$$



Rysunek 4.4: Odpowiedni podział z maksymalnym marginesem

4.8.1 Rozwiązanie dla problemów liniowo separowalnych

Funkcją klasyfikującą jest hiperpłaszczyzna będąca rozwiązaniem problemu optymalizacyjnego polegającego na zminimalizowaniu formy

$$Q(\mathbf{w}, b) = \frac{1}{2}||\mathbf{w}||^2 \tag{4.31}$$

przy warunkach

$$y_i(\mathbf{w} \cdot x_i + b) \geq 1 \text{ dla } 1 \leq i \leq N. \tag{4.32}$$

Minimalizacja daje hiperpłaszczyznę o maksymalnym marginesie.

4.8.2 Maksymalizacja marginesu hiperpłaszczyzny decyzyjnej

Wprowadzamy lagrangian i mnożniki Lagrange'a $\boldsymbol{\alpha}$ [33, 15, 1, 9, 14] dla którego rozwiązaniem będzie $(\mathbf{w}, b)$ maksymalizujące wyrażenie:

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{i=N} \alpha_i[y_i(\mathbf{w} \cdot x_i + b) - 1],\ \alpha_i \geq 0. \tag{4.33}$$

Lagrangian musi być minimalizowany ze względu na $\mathbf{w}$ i b, oraz maksymalizowany ze względu na $\boldsymbol{\alpha}$. Punkty dla $\alpha_i \geq 0$ są wektorami podpierającymi (ang. support vectors). Oznacza to, że punkt siodłowy funkcji $L(\mathbf{w}, b, \boldsymbol{\alpha})$ ma zostać znaleziony.

4.8.3 Zadanie dualne i warunki Kuhna-Tuckera

Warunki Kuhna-Tuckera. Dane służą do określenia punktu siodłowego w postaci

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial \mathbf{w}} = 0 \tag{4.34}$$

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial b} = 0. \tag{4.35}$$

Dla równania 4.33 przy wykorzystaniu warunków Kuhna-Tuckera otrzymamy wzory 4.36 i 4.37

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i x_i y_i \tag{4.36}$$

$$\sum_{i=1}^{i=N} \alpha_i y_i = 0. \tag{4.37}$$

Podstawiamy do funkcji Lagrange'a ograniczenie w problemie dualnym czyli mamy zadanie dualne. Zadanie dualne polega na maksymalizacji

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{i=N} \alpha_i - \frac{1}{2} \sum_{i,j}^{N} \alpha_i \alpha_j y_i y_j (x_i x_j) \tag{4.38}$$

przy ograniczeniach:

$$\alpha_i \geq 0,\ dla\, i = 1, 2, ..., N,\ oraz \sum_{i=1}^{i=N} \alpha_i y_i = 0$$

gdzie α_i zależy od x_i tylko poprzez iloczyny skalarne $x_i x_j$.

Rozwiązania $\boldsymbol{\alpha}, (\mathbf{w}, b)$ spełniają równanie:

$$\alpha_i[c_i(\mathbf{w} \cdot x_i + b) - 1] = 0 \tag{4.39}$$

natomiast

$$\alpha_i = 0, \qquad y_i(\mathbf{w} \cdot x_i + b) < 1 \tag{4.40}$$

x_i wewnątrz zbioru dopuszczalnego

$$\alpha_i \geq 0, \qquad y_i(\mathbf{w} \cdot x_i + b) = 1. \tag{4.41}$$

4.8.4 Wyznaczenie parametru b

Parametr b można wyznaczyć z ograniczeń pierwotnych:

$$b = -\frac{1}{2}(\min_i w_i \cdot x_i + \max_i w_i \cdot x_i) \tag{4.42}$$

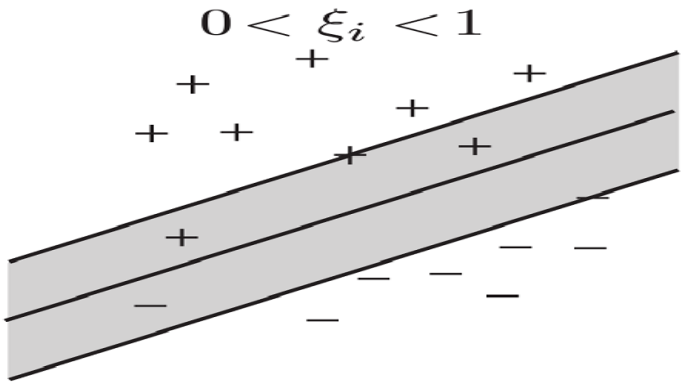
lub dla dowolnego wektora podpierającego $y_j(\mathbf{w} \cdot x_j + b) = 1$.

4.8.5 Rozwiązanie dla problemów liniowo nieseparowalnych

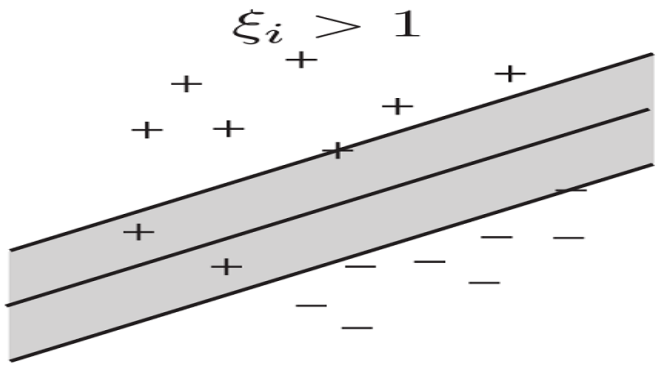
Problemy liniowo nieseparowalne rozwiązujemy wprowadzając zmienne rozluźniające ξ_i (ang. slack variables).

- $\xi_i = 0 - x_i$ sklasyfikowany poprawnie
- $\xi_i =$ odległość $-x_i$ sklasyfikowany niepoprawnie

$$y_i(w_i \cdot x_i - b) \geq 1 - \xi_i \tag{4.43}$$



Rysunek 4.5: Naruszony region rozdzielający ($0 < \xi_i < 1$)



Rysunek 4.6: Brak separacji liniowej ($\xi_i > 1$)

Rozwiązanie problemu liniowo nieseparowalnego polega na minimalizacji:

$$\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{i=N}\xi_i, \tag{4.44}$$

gdzie C jest arbitralnym parametrem; przy warunkach:

$$y_i(w_i \cdot x_i - b) \geq 1 - \xi_i, \qquad \xi_i \geq 0. \tag{4.45}$$

Funkcja Lagrange'a dla danego problemu wyraża się w następujący sposób:

$$L(\mathbf{w}, b, \xi, \boldsymbol{\alpha}) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{i=N}\xi_i \sum_{i=1}^{i=N}\alpha_i[1 - x_i - y_i(w_i \cdot x_i + b)] - \sum_{i=1}^{i=N}\lambda_i\xi_i. \tag{4.46}$$

Funkcja dualna identyczna jak w poprzednim przypadku:

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{i=N}\alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j k(x_i, x_j) \tag{4.47}$$

przy ograniczeniach:

$$\alpha_i \geq 0, \qquad \sum_{i=1}^{i=N}\alpha_i y_i = 0. \tag{4.48}$$

Dla przykładu, jądro wielowymiarowe w postaci

$$k(x_i, x_j) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)(y_1^2, y_2^2, \sqrt{2}y_1y_2)^T = (\Phi(x) \cdot \Phi(x')) \tag{4.49}$$

jest iloczynem skalarnym gdy

$$\Phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2) \tag{4.50}$$

$k(x_i, x_j)$ jest jądrem Mercer'a.
Ogólnie

$$k(\mathbf{x}, \mathbf{z}) = (\mathbf{x} \cdot \mathbf{z})^n = \sum_{i_1=1}^{n} ... \sum_{i_n=1}^{n}(x_{i_1}...x_{i_n})(z_{i_1}...z_{i_n}). \tag{4.51}$$

Wartości mnożników klasyfikują punkty x_i:

$$\alpha_i = 0, \qquad y_i(w_i \cdot x_i + b) < 1 \tag{4.52}$$

x_i wewnątrz zbioru dopuszczalnego

$$0 \leq \alpha_i \leq y_i, \qquad c_i(w_i \cdot x_i + b) = 1 \tag{4.53}$$

x_i jest obrazem podpierającym

$$\alpha_i = y_i, \qquad y_i(w_i \cdot x_i + b) = 1 \tag{4.54}$$

x_i narusza ograniczenia

$$\xi_i > 0$$

Rozwiązaniem zagadnienia optymalizacji jest

$$\mathbf{w} = \sum_{i=1}^{i=N} \alpha_i y_i x_i. \tag{4.55}$$

Rozwiązania $\boldsymbol{\alpha}, (\mathbf{w}, b)$ spełniają równanie

$$\alpha_i[y_i(w_i \cdot x_i + b) - 1 + \xi_i] = 0, \qquad (C - \alpha_i)\xi_i = 0. \tag{4.56}$$

4.9 Algorytm Lagrange'a

Algorytm bazuje na dualnej formie (wzór 4.38). Po odpowiednich przekształceniach sprowadza się do iteracyjnego schematu opisanego poniżej

$$x^{i+1} = Q^{-1}(\mathrm{e} + ((Qx^i - \mathrm{e}) - \alpha \mathrm{x}^i)), \qquad i = 0, 1, \ldots \tag{4.57}$$

realizowanego aż do momentu spełnienia warunku stopu

$$0 \leq \alpha \leq \frac{2}{\nu}. \tag{4.58}$$

Przyjmijmy pomocnicze oznaczenia:

$$H = D[A - \mathrm{e}], \qquad Q = \frac{1}{\nu} I + HH^T \tag{4.59}$$

gdzie $A \in R^{N \times M}$ macierz danych wejściowych, D macierz diagonalna składająca się z wartości 1 oraz -1 odpowiednio dla punktów należących do jednej z klas, ν jest dodatnim parametrem, e to macierz o wymiarze $M \times 1$ składająca się z jedynek.

4.10 Przykład klasyfikacji pacjentów

Przykład klasyfikacji przeprowadzono dla pacjentów z pierwszego dnia hospitalizacji i dla 24 godziny hospitalizacji z masą urodzeniową do 1500 [g] ze wsparciem oddechowym – IMV/SIMV

Dane pacjentów zamieszczono w tablicy 4.1:

zgon do 14 doby	pO_2/FiO_2	pCO_2	pO_2	FiO_2
1	91,96	51,97	57,70	0,62
-1	70,15	93,44	56,73	0,83
-1	90,29	63,61	46,67	0,52
-1	39,74	72,48	38,18	0,97
1	92,60	41,34	49,01	0,56
1	76,26	50,09	54,18	0,64
1	57,48	48,41	51,59	0,92
-1	93,45	48,54	56,79	0,59
-1	59,32	68,48	56,52	0,97
-1	48,59	92,52	45,97	0,95
1	63,61	59,12	36,98	0,59
1	89,66	70,00	52,69	0,64
1	94,95	38,05	40,87	0,43
1	99,90	38,67	47,19	0,47
1	89,21	51,59	45,12	0,50
-1	54,79	64,54	44,13	0,88
1	91,03	63,95	50,83	0,57
-1	38,67	75,54	32,62	0,84
1	81,26	49,19	43,04	0,61
1	81,33	46,54	47,51	0,58
-1	79,47	65,82	51,65	0,66
1	91,85	58,41	39,75	0,47
1	89,00	43,11	49,99	0,56
1	93,47	44,45	43,95	0,46
1	99,83	40,34	62,21	0,62
1	62,13	67,18	40,78	0,66
-1	79,46	67,43	61,29	0,81
1	86,60	30,84	46,86	0,55
-1	75,83	50,77	63,27	0,87
1	99,14	39,06	79,58	0,83
1	64,62	45,46	55,68	0,87
1	73,90	59,39	45,59	0,62
1	69,92	50,18	48,65	0,75
1	74,73	52,28	39,71	0,60
1	91,07	42,49	90,08	0,99
1	62,18	53,89	56,44	0,90
1	93,60	42,52	90,59	0,96
1	82,18	38,38	69,42	0,80
1	83,72	47,79	37,33	0,45
1	73,83	69,81	54,67	0,73
1	66,45	62,10	41,63	0,70

Tablica 4.1: Dane pacjentów 24 godzina hospitalizacji

Dane pacjentów testowych zamieszczono w tablicy 4.2.

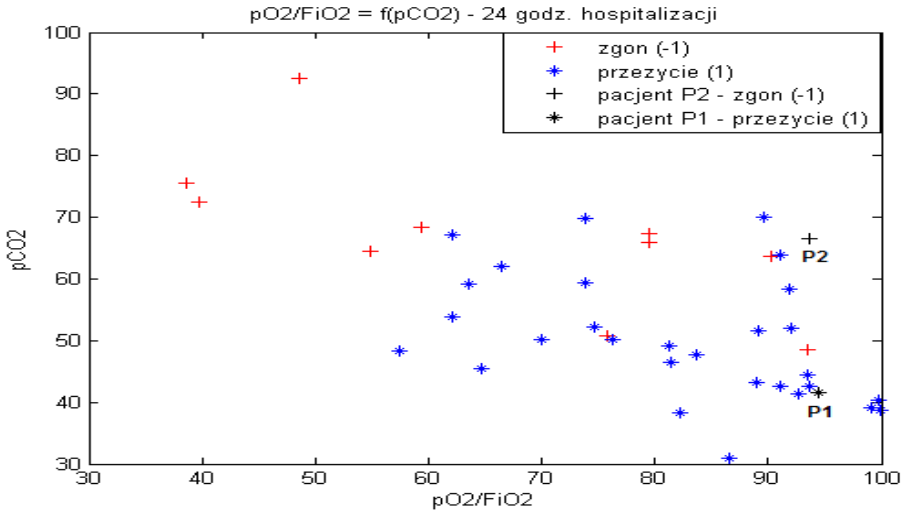
Oznaczenie	**zgon do 14 doby**	pO_2/FiO_2	pCO_2	pO_2	FiO_2
P1	1	94,38	41,46	37,74	0,39
P2	-1	93,68	66,59	52,59	0,58

Tablica 4.2: Dane pacjentów testowych 24 godzina hospitalizacji

Pacjenci testowi zostali oznaczeni jako P1 gdy nie wystąpił zgon do 14 doby życia oraz P2 gdy wystąpił zgon do 14 doby życia.

Przykład klasyfikacji dla parametrów wejściowych:
- zgon do 14 doby życia kodowany jako (-1 oznacza zgon, 1 oznacza przeżycie)
- parametr pO_2/FiO_2,
- parametr pCO_2.

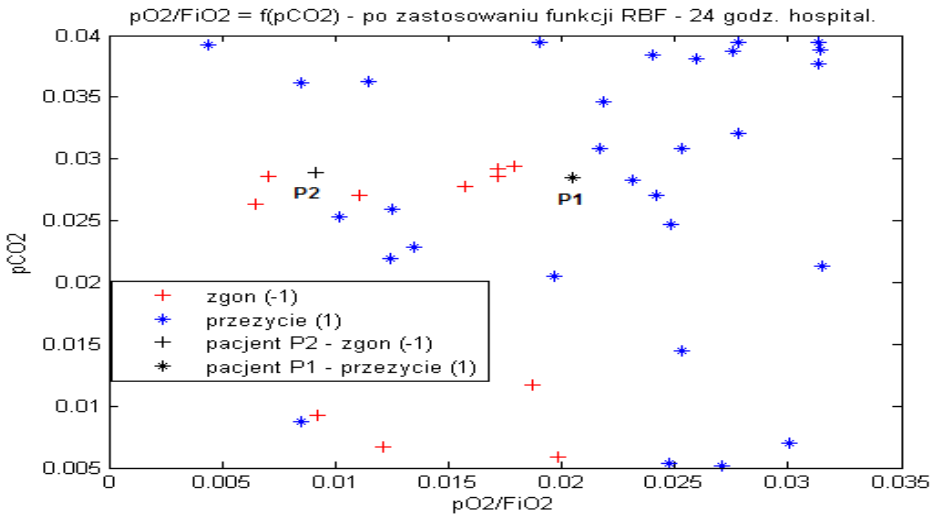
Rysunek 4.7 przedstawia zależność $pO_2/FiO_2 = f(pCO_2)$ dla danych pacjentów.



Rysunek 4.7: Oryginalna przestrzeń klasyfikacji danych

Dla wszystkich danych pacjentów zastosowano funkcję jądra typu RBF.

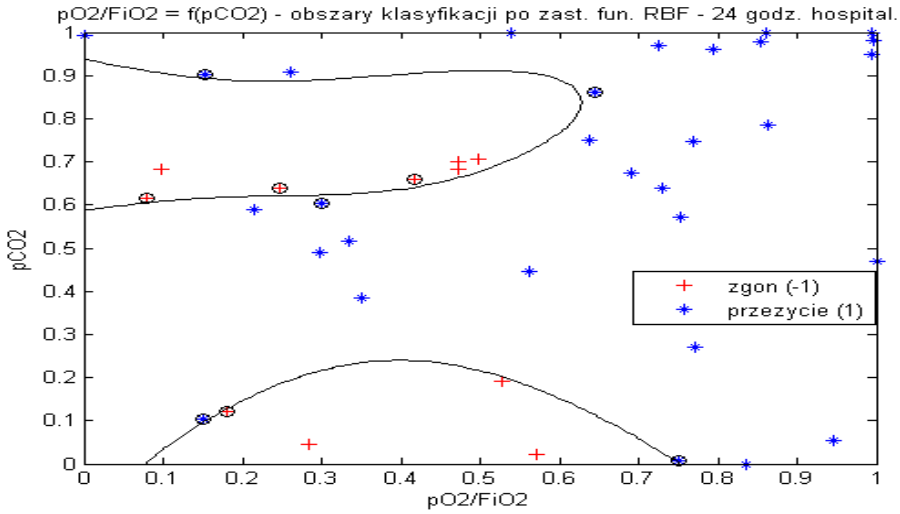
Rysunek 4.8 przedstawia zależność $pO_2/FiO_2 = f(pCO_2)$ dla danych pacjentów po zastosowaniu funkcji jądra typu RBF.



Rysunek 4.8: Klasyfikacja danych po zastosowaniu radialnej funkcji jądrowej Φ

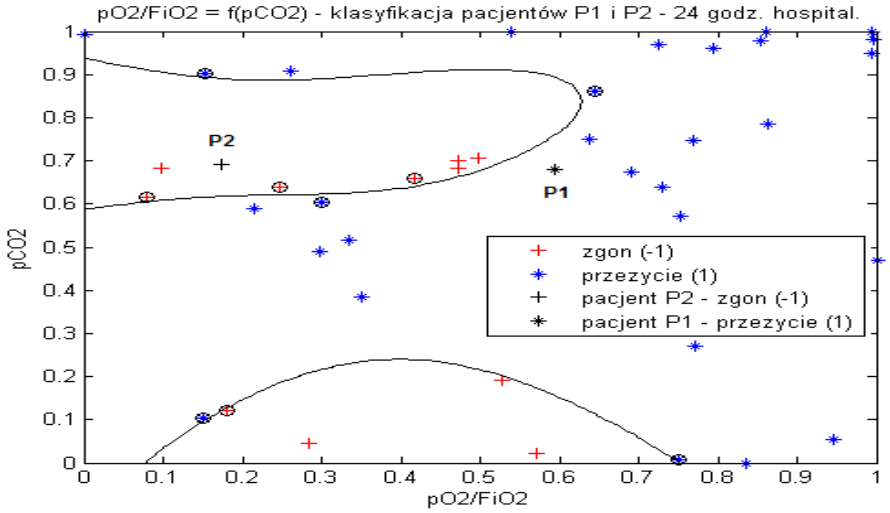
Zaznaczono obszary klasyfikacji po zastosowaniu funkcji jądra typu RBF bez uwzględnienia pacjentów testowych P1 i P2.

Rysunek 4.9 przedstawia zależność $pO_2/FiO_2 = f(pCO_2)$ dla danych pacjentów z zaznaczonymi obszarami klasyfikacji.



Rysunek 4.9: Krzywe rozdzielające przypadki

Przeprowadzono test klasyfikacji pacjentów testowych P1 i P2.
Rysunek 4.10 przedstawia wyniki klasyfikacji dla dwóch pacjentów testowych.

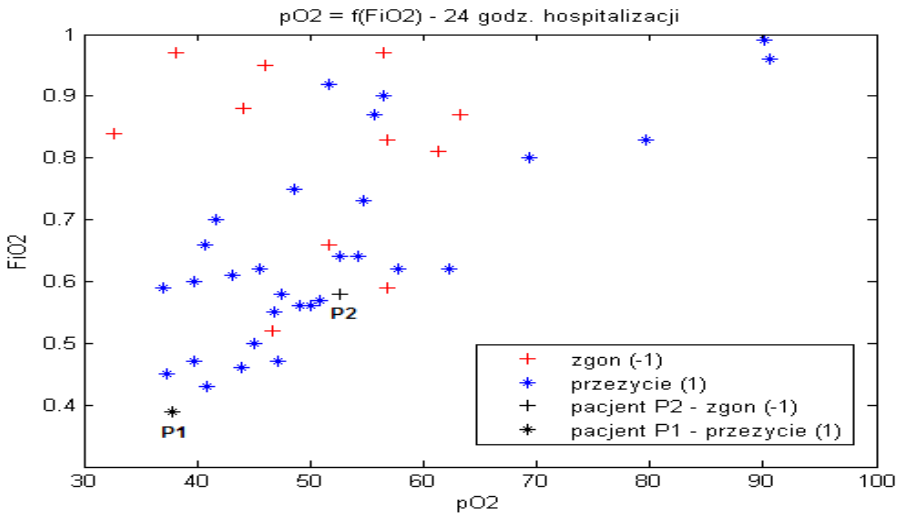


Rysunek 4.10: Test dla danych P1 oraz P2

W tym przykładzie klasyfikacji Pacjent P2 został zaklasyfikowany do grupy pacjentów, u których wystąpił zgon do 14 doby życia, natomiast pacjent P1 został zaklasyfikowany do grupy pacjentów, u których nie wystąpił zgon do 14 doby życia. Metoda SVM poprawnie zaklasyfikowała pacjentów testowych P1 i P2.

Przykład klasyfikacji dla parametrów wejściowych:
- zgon do 14 doby życia kodowany jako (-1 oznacza zgon, 1 oznacza przeżycie)
- parametr pO_2,
- parametr FiO_2

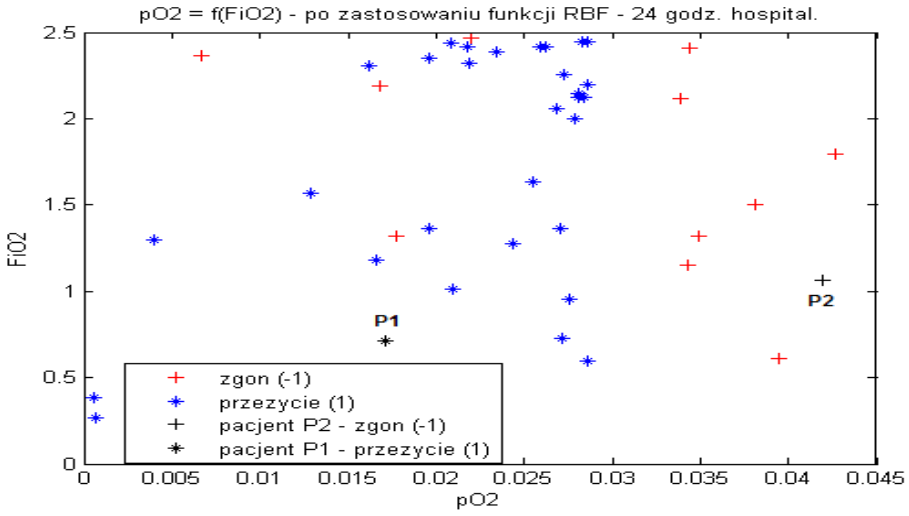
Rysunek 4.11 przedstawia zależność $pO_2 = f(FiO_2)$ dla danych pacjentów.



Rysunek 4.11: Oryginalna przestrzeń klasyfikacji danych

Dla wszystkich danych pacjentów zastosowano funkcję jądra typu RBF.

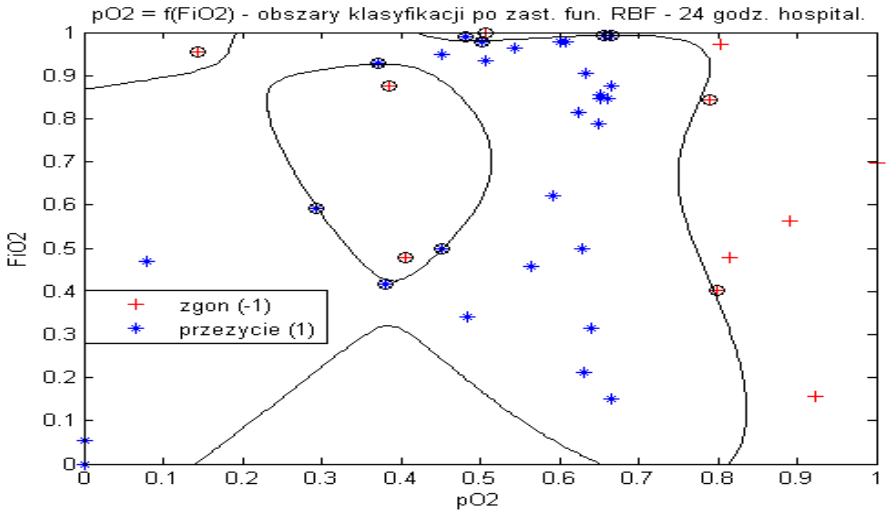
Rysunek 4.12 przedstawia zależność $pO_2 = f(FiO_2)$ dla danych pacjentów po zastosowaniu funkcji jądra typu RBF.



Rysunek 4.12: Klasyfikacja danych po zastosowaniu radialnej funkcji jądrowej Φ

Zaznaczono obszary klasyfikacji po zastosowaniu funkcji jądra typu RBF bez uwzględnienia pacjentów testowych P1 i P2.

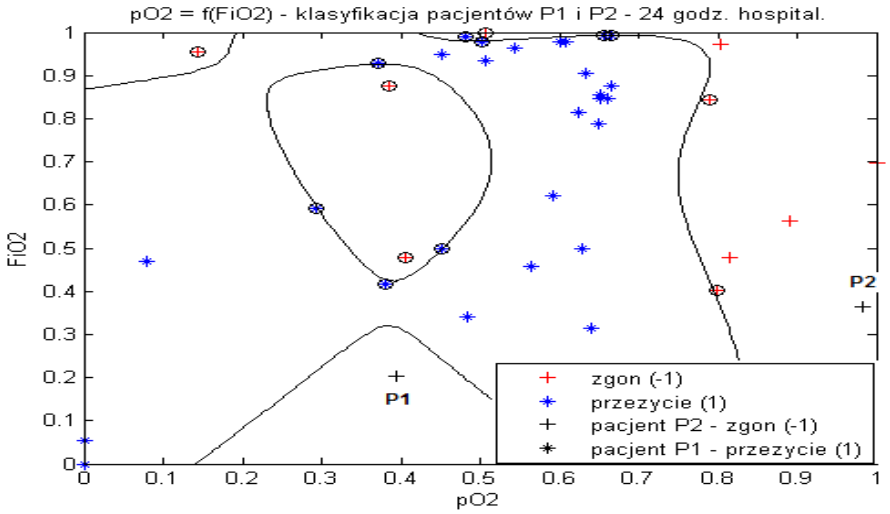
Rysunek 4.13 przedstawia zależność $pO_2 = f(FiO_2)$ dla danych pacjentów z zaznaczonymi obszarami klasyfikacji.



Rysunek 4.13: Krzywe rozdzielające przypadki

Przeprowadzono test klasyfikacji pacjentów testowych P1 i P2.

Rysunek 4.14 przedstawia wyniki klasyfikacji dla dwóch pacjentów testowych.



Rysunek 4.14: Test dla danych P1 oraz P2

W tym przykładzie klasyfikacji pacjenci P1 i P2 zostali zaklasyfikowani do grupy pacjentów, u których wystąpił zgon do 14 doby życia. Metoda SVM poprawnie zaklasyfikowała pacjenta testowego P2, natomiast pacjent P1 nie został poprawnie zaklasyfikowany.

Rozdział 5

Model i algorytm postępowania lekarza

5.1 Predykcja wskaźnika pO_2/FiO_2 przy użyciu modelu postępowania lekarza (MPL)

Predykcja wskaźnika pO_2/FiO_2 przy użyciu modelu symulacji jest celem tej części pracy. W uproszczeniu, działanie systemu umożliwia aktywne modelowanie przestrzeni zdarzeń poprzez zadawanie liczby i wartości parametrów w zadanych terminach modelowanego przebiegu hospitalizacji. Model pozwala ocenić wpływ modelowanych decyzji na przebieg obserwowanego procesu chorobowego. Zasadnicza różnica pomiędzy proponowanymi wcześniej metodami AIS oraz SVM, a prezentowaną metodą polega na możliwości aktywnego kształtowania wartości parametrów w przyszłych, przewidywanych stanach chorobowych. Model umożliwia ocenę decyzji lekarza poprzez autoocenę własnych decyzji zgodnie z przyjętym przez lekarza kryterium jakości przebiegu procesu w modelowanej metodami matematycznymi przestrzeni przewidywanych zdarzeń.

Prezentowany program umożliwia wyszukanie pacjentów o podobnym przebiegu powikłań oddechowych w dostępnej bazie danych. Przypadek podobny może być określony arbitralnie. Wybór arbitralny dotyczy liczby parametrów, które opisują dany przypadek chorobowy. Ilość i rodzaj parametrów może ulegać zmianie. Wynika to po części ze zmiany poglądów na temat istotności tych parametrów na dokładność opisu przypadku chorobowego, a po części z pojawieniem się nowych wielkości pomiarowych w związku z rozwojem technik pomiarowych, a także dostępności nowych technik diagnozowania. Problem podobieństwa jest tu rozpatrywany w kategoriach statystycznych. Na przykład, gdy dana wielkość ma rozkład normalny, można przyjąć miarę różnicującą odległość w postaci wartości odchylenia standardowego wybranego parametru. W przypadku zmiennych przyjmujących dwie wartości, podobieństwo jest łatwe do określenia. Czynnikami kwalifikującymi przypadek jako przypadek podobny do zadanego wzorca są wartości pomiarów gazometrii rozrzucone statystycznie w pobliżu wartości zadanych, które opisują przypadek chorobowy pacjenta. Program umożliwia analizę wartości parametrów gazometrii: pO_2/FiO_2, pH, pCO_2, HCO_3.

Wartości parametrów opisanych jako pO_2/FiO_2, pH, pCO_2, HCO_3 podawane są w parach łącznie z terminem ich pomiaru. Termin pomiaru wyznaczany jest od terminu przyjęcia pacjenta na oddział kliniczny. Ten termin określony jest wartością zero. Inne parametry kodowane jako zero lub jeden dotyczą faktu wystąpienia lub nie wystąpienia określonych chorób, lub stanów nienormalnych jak: odma, posocznica, IVH, PDA, NEC, fakt podania surfaktantu w zadanym przedziale czasu, oraz wiek płodowy wyrażony w tygodniach. Zadaniem metody poszukiwania przypadków chorobowych, które stanowią zbiór przypadków podobnych, jest umożliwienie lekarzowi kreującemu kryteria podobieństwa, wyszukać w bazie danych przypadki podobne do aktualnie modelowanego.

Wynikiem działania programu napisanego w oparciu o metodę poszukiwań jest wykres prezentujący przewidywany przebieg wskaźnika pO_2/FiO_2 oraz tabele wygenerowane dla każdego podobnego przypadku chorobowego zapisanego w bazie danych.

Predykowany przebieg wskaźnika pO_2/FiO_2 obliczany jest jako wartość średnia ze wszystkich przebiegów tego wskaźnika i dla pacjentów uznanych jako przypadki podobne do rozważanego w systemie modelowania.

Predykcja wskaźnika pO_2/FiO_2 przy pomocy przedstawionej powyżej aplikacji polega na wpisywaniu w czasie rzeczywistym dostępnych wyników pomiarów gazometrii poprzez komputerowe interfejsy pomiarowe do bazy danych, szczególnie wskaźnika pO_2/FiO_2, zaznaczeniu rozpoznanych chorób mających szczególny wpływ na przebieg powikłań oddechowych pacjenta wykrytych w dowolnym okresie dotychczasowej hospitalizacji, zaznaczeniu podania surfaktantu w pierwszej i drugiej dobie hospitalizacji, gdy taka sytuacja miała miejsce. Funkcjonalność ta ma zastosowanie nie tylko do dokładniejszego dopasowania podobnych przebiegów, ale także umożliwia zobrazowanie przewidywanego przebiegu wskaźnika pO_2/FiO_2 w przypadku zamiaru podjęcia decyzji o podaniu surfaktantu, lub o zrezygnowaniu z podania leku.

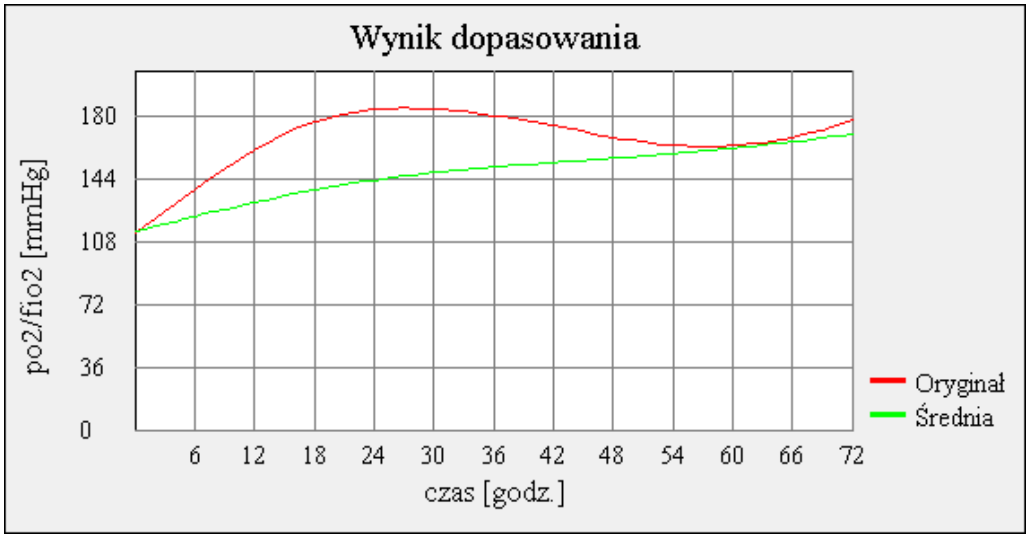
Na rysunkach przedstawiono przykład analizy powikłań oddechowych. Na wykresie kolorem czerwonym przedstawiony jest rzeczywisty przebieg wskaźnika pO_2/FiO_2 dla testowego pacjenta. Kolorem zielonym przedstawiono przewidywany przebieg tego wskaźnika w drodze modelowania.

5.1.1 Predykcja wskaźnika pO_2/FiO_2 - przykład 1

Predykcja wskaźnika pO_2/FiO_2 została przeprowadzona dla pacjenta z masą urodzeniową < 1500 [g].

Pacjent wymagał wentylacji przy użyciu respiratora IMV lub SIMV. Lekarz poszukuje w bazie danych przypadków podobnych do rozpatrywanego aktualnie przypadku chorobowego poprzez wprowadzanie wartości wskaźnika $pO_2/FiO_2 = 113.685$ i $pH = 7.32$ w terminie określonym jako godzina zero, patrz rysunek 5.1, uzyskując wstępną prognozę, patrz rysunek 5.2 i rysunek 5.3.

Rysunek 5.1: Dane wprowadzone dla godziny 0

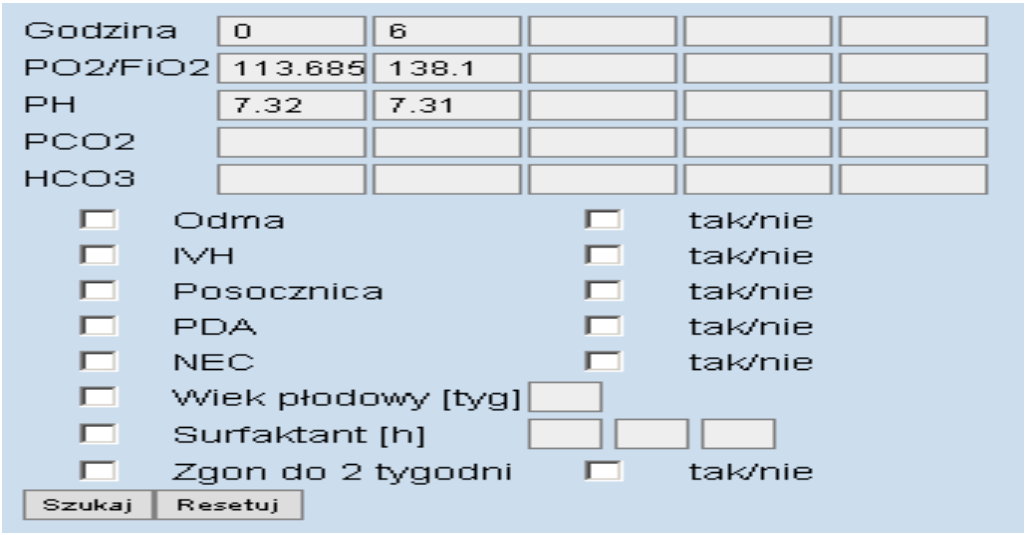


Rysunek 5.2: Wynik predykcji wskaźnika pO_2/FiO_2 od godziny 6 do 72

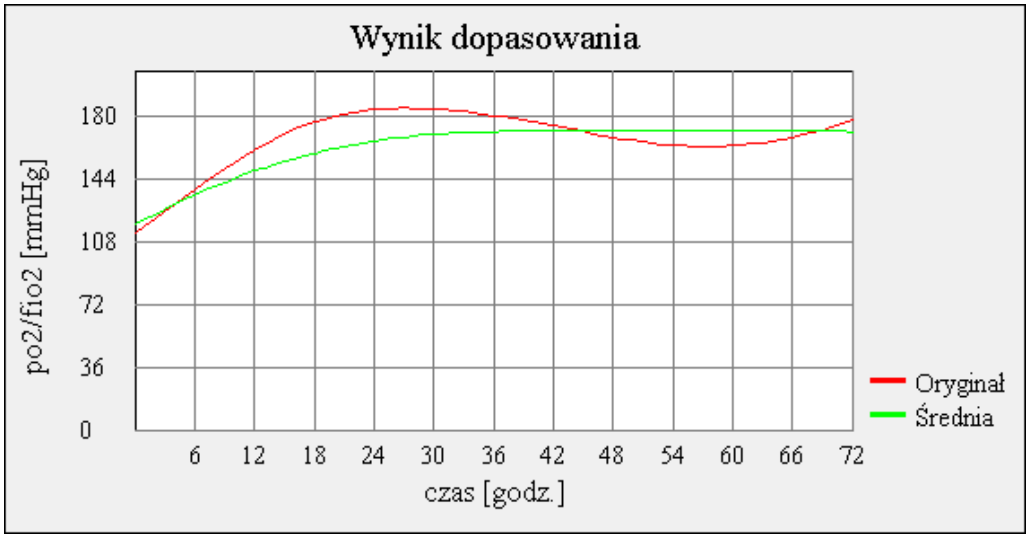
Wiek pł.	25 tygodni												
Surf.	0h, 8h												
Zgon	nie wystąpił w okresie hospitalizacji												
													Dane dopasowane
Godzina	0	6	12	18	24	30	36	42	48	54	60	66	72
PO2/FiO2	113.8	134.9	152.3	167.8	180.9	189.3	191	188	182.9	177.7	173.4	171	171.3
PH	7.36	7.34	7.32	7.3	7.29	7.28	7.28	7.28	7.28	7.29	7.29	7.29	7.28
PCO2	33.1	35.68	38.21	40.33	41.59	41.69	41.28	40.39	39.08	37.58	36.3	35.61	35.23
HCO3	18.31	18.6	18.89	19.05	19.02	18.78	18.47	18.11	17.68	17.24	16.87	16.61	16.35
FiO2	0.47	0.4	0.35	0.32	0.3	0.29	0.29	0.29	0.29	0.29	0.29	0.29	0.28
Odma													
IVH													
Poso.													
PDA	pda	pda	pda	pda	pda	pda	pda	pda	pda	pda			
NEC													
Wiek pł.	24 tygodni												
Surf.	0h, 7h												
Zgon	nie wystąpił w okresie hospitalizacji												
													Dane dopasowane
Godzina	0	6	12	18	24	30	36	42	48	54	60	66	72
PO2/FiO2	115.3	101.1	91.02	86.93	89	95.06	101.1	106.4	113.7	123.9	133.9	142.2	149.1
PH	7.28	7.3	7.32	7.32	7.32	7.31	7.29	7.28	7.28	7.27	7.27	7.27	7.27
PCO2	50.43	46.77	44.24	43.1	43.11	44.01	45.59	47.49	49.13	50.25	51.1	51.8	52.31
HCO3	22.44	22.23	22.11	22.04	21.98	21.93	22.01	22.32	22.69	22.99	23.26	23.49	23.64
FiO2	0.39	0.44	0.48	0.52	0.56	0.57	0.57	0.57	0.57	0.56	0.55	0.53	0.51
Odma													
IVH													
Poso.													
PDA													

Rysunek 5.3: Podgląd danych z dopasowanych przypadków od godziny 0 do 72

Następnie lekarz wprowadza wartości wskaźnika $pO_2/FiO_2 = 138.1$ i $pH = 7.31$ w terminie określonym jako godzina szósta, patrz rysunek 5.4, uzyskując wyniki jak na rysunku 5.5.

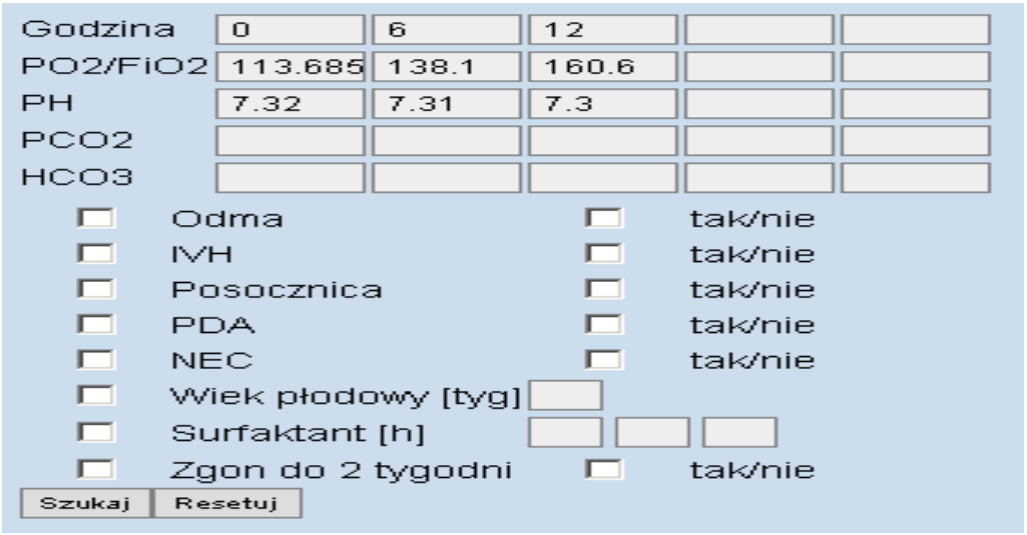


Rysunek 5.4: Dane wprowadzone dla godziny 0 i 6

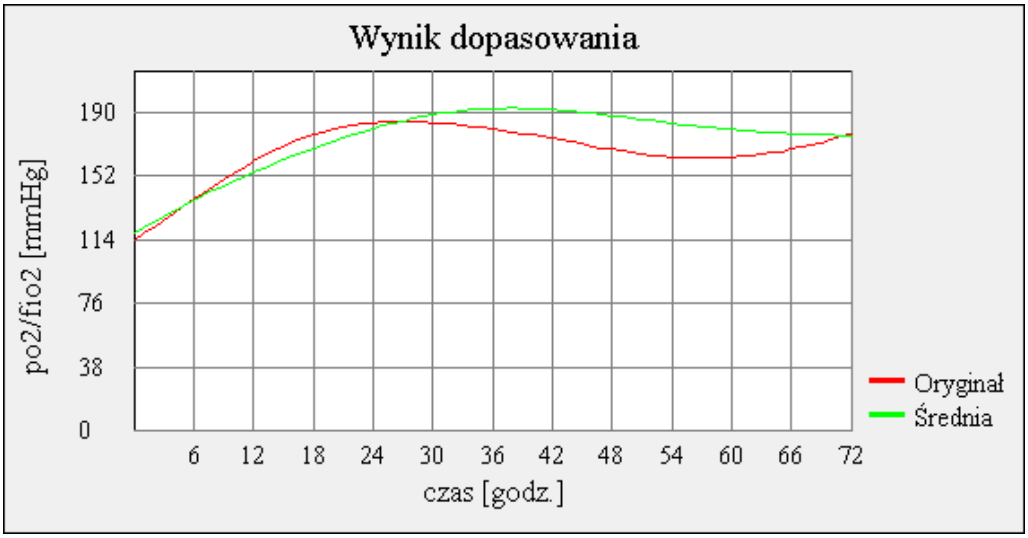


Rysunek 5.5: Wynik predykcji wskaźnika pO_2/FiO_2 od godziny 12 do 72

Lekarz wprowadza wartości wskaźnika $pO_2/FiO_2 = 160.6$ i $pH = 7.3$ w terminie określonym jako godzina dwunasta, patrz rysunek 5.6, uzyskując wyniki jak na rysunku 5.7.



Rysunek 5.6: Dane wprowadzone dla godziny 0, 6 i 12

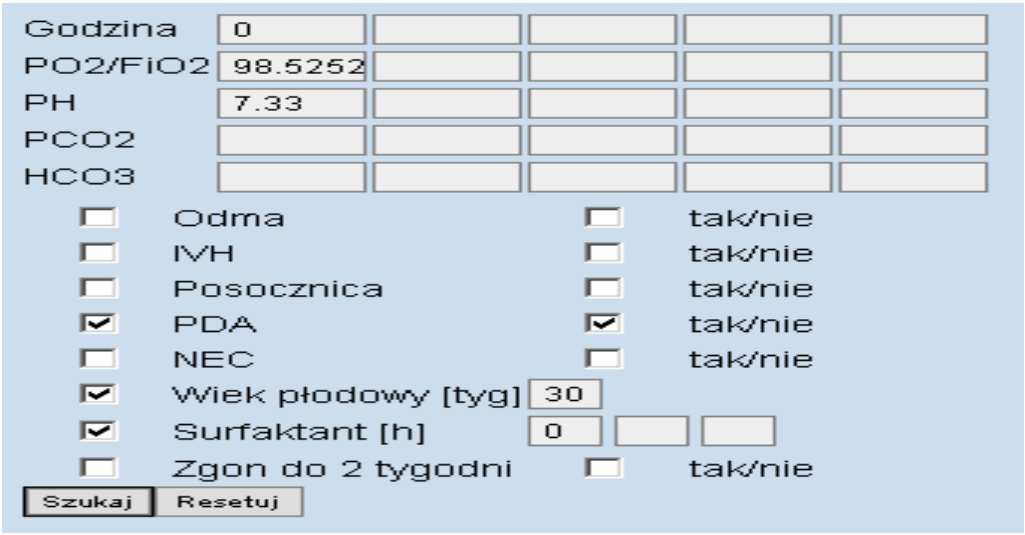


Rysunek 5.7: Wynik predykcji wskaźnika pO_2/FiO_2 od godziny 18 do 72

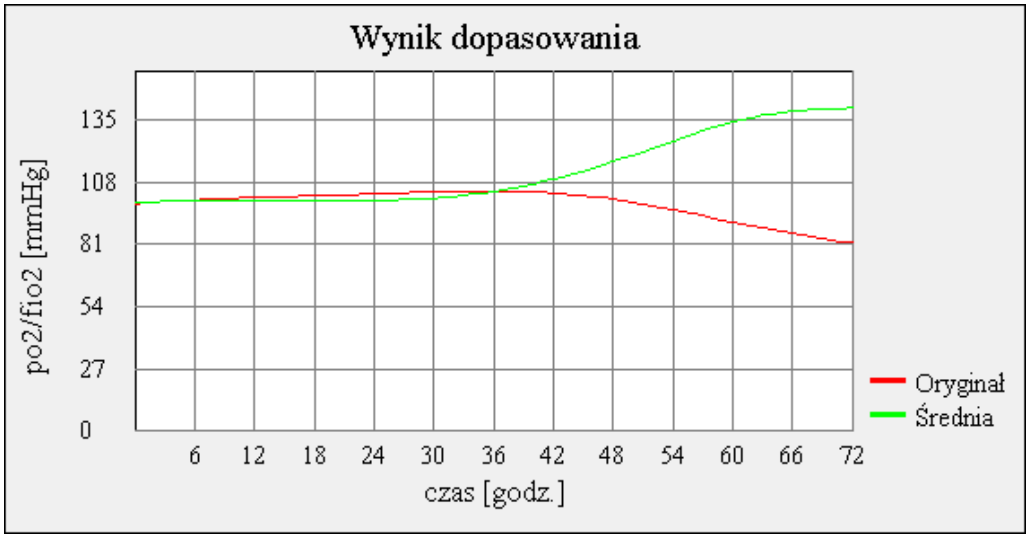
5.1.2 Predykcja wskaźnika pO_2/FiO_2 - przykład 2

Modelowanie wskaźnika pO_2/FiO_2 zostało przeprowadzone dla pacjenta z masą urodzeniową < 1500 [g].

Pacjent wymagał wentylacji przy użyciu respiratora IMV lub SIMV. Lekarz poszukujący w bazie danych przypadków podobnych do zadanego przypadku wprowadza wartości wskaźnika $pO_2/FiO_2 = 98.5252$ i $pH = 7.33$ w terminie określonym jako godzina zero oraz wprowadza rozpoznanie PDA. Lekarz zadaje termin podania surfaktantu w terminie określonym jako godzina zero oraz wiek płodowy pacjenta dla wartości 30 tygodni, patrz rysunek 5.8, a system generuje wstępną prognozę, patrz rysunek 5.9 i rysunek 5.10.



Rysunek 5.8: Dane wprowadzone dla godziny 0

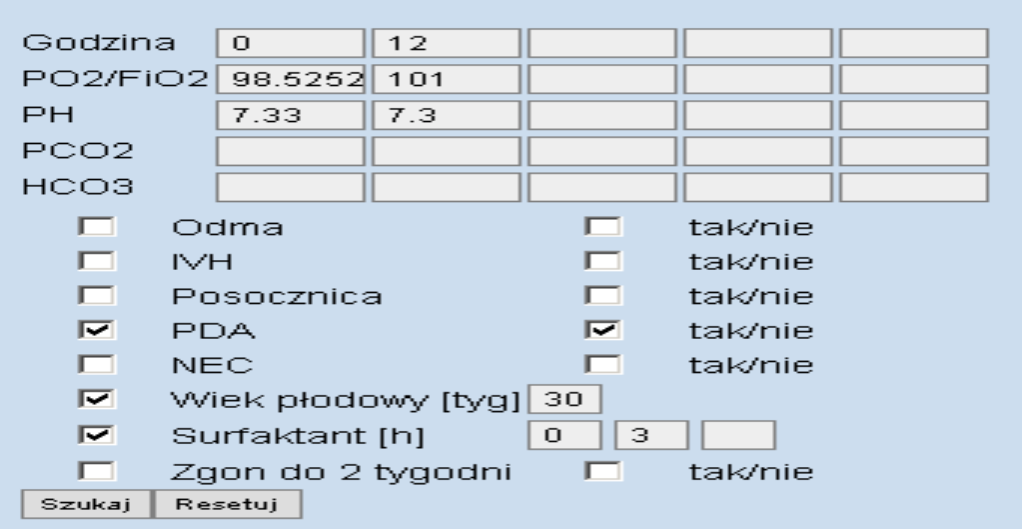


Rysunek 5.9: Wynik predykcji wskaźnika pO_2/FiO_2 od godziny 6 do 72

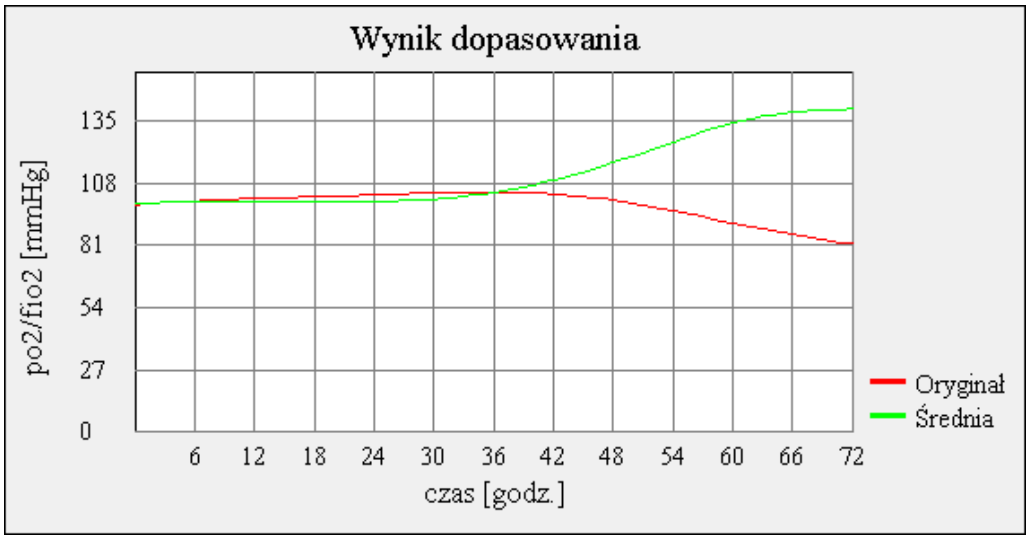
													Dane dopasowane
Godzina	0	6	12	18	24	30	36	42	48	54	60	66	72
PO2/FiO2	99.25	99.76	99.78	99.66	99.9	101	103.8	109.1	116.7	125.7	133.9	138.6	139.9
PH	7.33	7.34	7.34	7.34	7.34	7.33	7.32	7.3	7.29	7.28	7.27	7.26	7.25
PCO2	33.36	34.69	36.05	37.37	38.67	40.01	41.39	42.77	44.12	45.41	46.7	48.13	49.79
HCO3	17.11	18.14	19.05	19.73	20.19	20.47	20.64	20.72	20.76	20.79	20.89	21.08	21.41
FiO2	0.6	0.56	0.52	0.49	0.47	0.45	0.43	0.42	0.41	0.4	0.4	0.39	0.39
Odma													
IVH													
Poso.													
PDA	pda	pda	pda	pda	pda	pda	pda	pda	pda	pda	pda	pda	pda
NEC													
Wiek pł.	29 tygodni												
Surf.	0h, 8h												
Zgon	nie wystąpił w okresie hospitalizacji												
													Dane testowe
Godzina	0	6	12	18	24	30	36	42	48	54	60	66	72
PO2/FiO2	98.53	100.2	101.3	102	102.8	103.7	104	103.2	100.5	95.95	90.6	85.56	81.74
PH	7.3	7.3	7.3	7.3	7.3	7.3	7.3	7.29	7.29	7.28	7.28	7.28	7.29
PCO2	47.72	47.3	46.93	46.67	46.56	46.54	46.47	46.33	46.38	46.59	46.55	46.19	46.04
HCO3	23.21	23.1	22.98	22.86	22.73	22.56	22.32	22	21.71	21.52	21.4	21.38	21.67
FiO2	0.74	0.63	0.53	0.45	0.4	0.37	0.36	0.38	0.41	0.44	0.47	0.48	0.48
Odma													
IVH													
Poso.													
PDA	pda	pda	pda	pda	pda	pda	pda	pda	pda				
NEC													
Wiek pł.	30 tygodni												
Surf.	0h, 3h												
Zgon	nie wystąpił w okresie hospitalizacji												

Rysunek 5.10: Podgląd danych z dopasowanych przypadków od godziny 0 do 72

Następnie lekarz wprowadza wartości wskaźnika $pO_2/FiO_2 = 101.3$ i $pH = 7.3$ w terminie określonym jako godzina dwunasta, patrz rysunek 5.11 oraz zadaje termin podania surfaktantu określony jako godzina trzecia. System zwraca modelowany przebieg w postaci wykresów jak to przedstawiono na rysunku 5.12.



Rysunek 5.11: Dane wprowadzone dla godziny 0 i 12



Rysunek 5.12: Wynik predykcji wskaźnika pO_2/FiO_2 od godziny 18 do 72

Rozdział 6

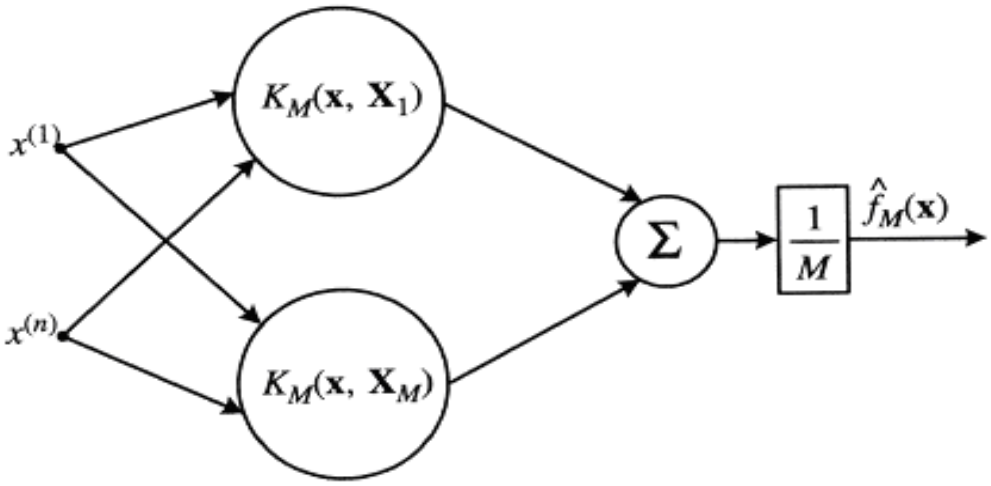
Sztuczne Sieci Neuronowe (SSN)

6.1 Probabilistyczne sieci neuronowe

Do prognozy ryzyka zgonu została wykorzystana probabilistyczna sieć neuronowa. Probabilistyczne sieci neuronowe zaprojektował Specht [17]. W tej pracy SSN są stosowane dla porównania wyników uzyskanych innymi metodami. Cechują się one wysoką skutecznością, ale mają pewne niedostatki. Te niedostatki wynikają w przeważającej części z trudności w wyciąganiu ogólniejszych wniosków z rezultatów, które te metody dostarczają. Koncepcja ich wywodzi się z nieparametrycznych metod estymacji funkcji gęstości i regresji. W wielu zagadnieniach identyfikacji, klasyfikacji i predykcji zachodzi potrzeba estymacji funkcji gęstości prawdopodobieństwa [22, 21, 16, 8]. W celu estymacji tej funkcji możemy zastosować estymator

$$\hat{f}_M(x) = \frac{1}{M}\sum_{i=1}^{M} K_M(x, X_i) \tag{6.1}$$

gdzie $X_1, \ldots, X_M$ jest ciągiem obserwacji n -wymiarowej zmiennej losowej X o gęstości prawdopodobieństwa f, natomiast K_M jest odpowiednio dobranym jądrem.



Rysunek 6.1: Probalistyczna sieć neuronowa - estymacja funkcji gęstości

Na rysunku 6.1 przedstawiono realizację sieciową estymatora (6.1). Należy zaznaczyć, że proponowana sieć nie wymaga procesu uczenia (optymalnego doboru wag połączeń), gdyż funkcję wag pełnią kolejne składowe wektorów obserwacji X_i. Jako funkcję K_M można przyjąć tzw. *jądro Parzena* postaci

$$K_M(x,u) = h_M^{-n} K(\frac{x-u}{h_M}) \tag{6.2}$$

przy czym ciąg h_M jest funkcją długości ciągu uczącego M i powinien spełniać warunki

$$\lim_{M\to\infty} h_M = 0 \quad i \quad \lim_{M\to\infty} M h_M^n = \infty \tag{6.3}$$

Można wykazać, że

$$E\left[\hat{f}_M(x) - f_M(x)\right]^2 \stackrel{M}{\to} 0. \tag{6.4}$$

w punktach ciągłości gęstości f. Funkcję K we wzorze (6.2) wygodnie jest przedstawić w postaci

$$K(x) = \prod_{i=1}^{n} H(x^{(i)}). \tag{6.5}$$

Zakładając, że funkcja H jest typu gaussowskiego, mamy

$$\hat{f}_M(x) = \frac{1}{(2\pi)^{\frac{n}{2}} n h_M^n} \sum_{i=1}^{M} \exp(-\frac{(x-X_i)^T(x-X_i)}{2h_M^2}). \tag{6.6}$$

W sposób analogiczny możemy skonstruować probalistyczną sieć neuronową w celu estymacji funkcji regresji.

Niech (X, Y) będą parą zmiennych losowych. Załóżmy, że X przyjmuje wartości w zbiorze R^n, natomiast Y w zbiorze R . Niech f będzie funkcją gęstości prawdopodobieństwa zmiennej losowej X . Na podstawie M niezależnych obserwacji $(X_1, Y_1), \ldots, (X_M, Y_M)$ zmiennych (X, Y) należy estymować funkcję regresji R zmiennej losowej Y względem X, tj.

$$\phi(x) = E\left[Y|X = x\right]. \tag{6.7}$$

Zdefiniujmy funkcję

$$R(x) = \phi(x) \cdot f(x). \tag{6.8}$$

Do estymacji funkcji (6.8) można zastosować estymator analogiczny do procedury (6.2), tzn.

$$\hat{R}_M(x) = \frac{1}{M}\sum_{i=1}^{M} Y_i K_M(x, X_i). \tag{6.9}$$

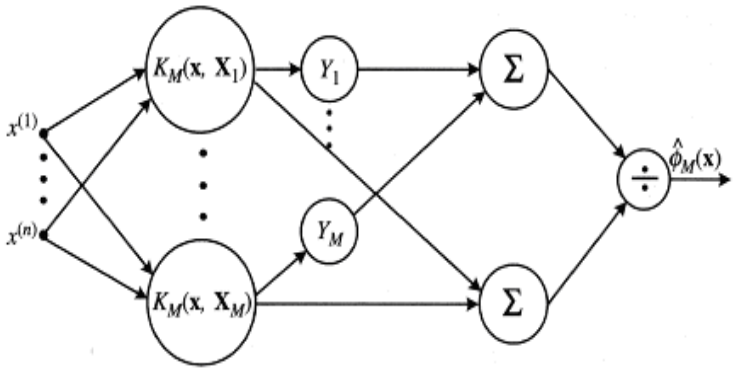
Zatem funkcję regresji estymujemy za pomocą

$$\hat{\phi}_M(x) = \frac{\hat{R}_M(x)}{\hat{f}_M(x)}. \tag{6.10}$$

W konsekwencji otrzymujemy estymator:

$$\hat{\phi}_M(x) = \frac{\sum_{i=1}^{M} Y_i K(\frac{x-X_i}{h_M})}{\sum_{i=1}^{M} K(\frac{x-X_i}{h_M})} \tag{6.11}$$

Na rysunku 6.2 pokazano realizację neuronową estymatora funkcji regresji nawiązującą w swojej konstrukcji do struktury danej na rysunku 6.1. Przedstawione struktury probabilistycznych sieci neuronowych nie wymagają uczenia. Ponadto zastosowanie tych sieci przy odpowiednio dobranym ciągu h_M gwarantuje zbieżność estymatorów. Są to wyniki asymptotyczne. W praktyce musimy zagwarantować posiadanie ciągów uczących o znacznej długości. Obie struktury można zastosować do rozwiązania zagadnienia klasyfikacji.



Rysunek 6.2: Probalistyczna sieć neuronowa do estymacji funkcji regresji

Rozdział 7

Wyniki AIS i MPL - Predykcja wskaźnika pO_2/FiO_2

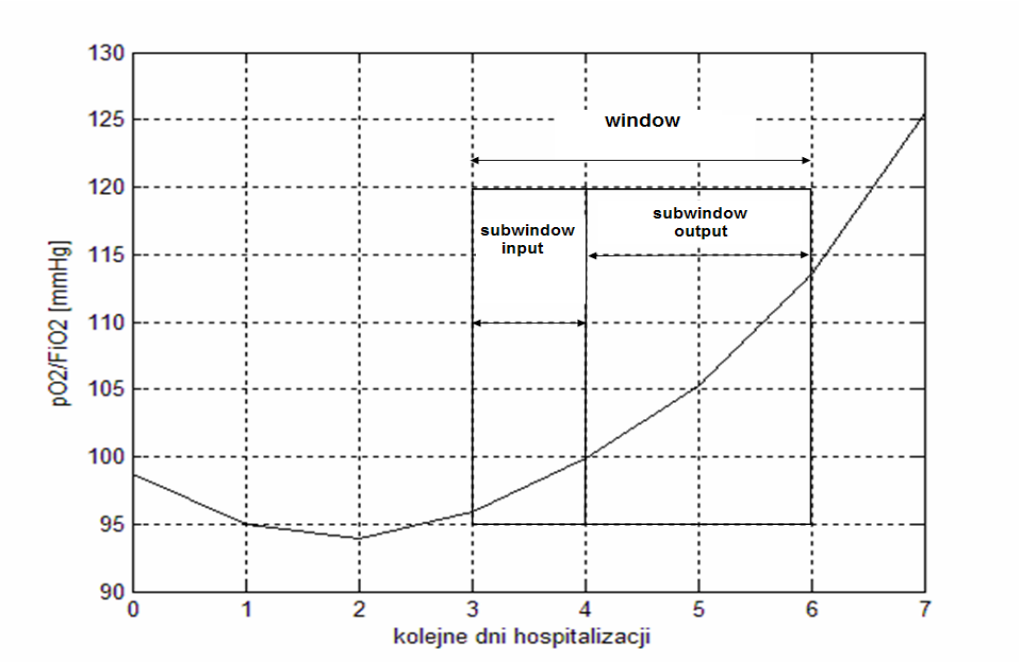
Proces uczenia sztucznej sieci immunologicznej (AIS) był realizowany w środowisku rozproszonym typu *klaster MATLAB Distributed Computing (MDC)*. W skład klastra wchodzą pakiety *MATLAB Distributed Computing Engine* oraz *MATLAB Distributed Computing Toolbox*. Proces predykcji został zrealizowany w środowisku MATLAB 7.1 (R14). Powyższe środowiska obliczeniowe zostały zainstalowane na serwerach obliczeniowych PWSZ w Krośnie. Funkcje użyte do uczenia i predykcji zostały zaprezentowane w dodatku A.

7.1 Predykcja wskaźnika pO_2/FiO_2 metodą sztucznej sieci immunologicznej (AIS)

Predykcję przeprowadzono zgodnie z algorytmem AIS nr 1 podanym w rozdziale 3. Sieć immunologiczna umożliwia predykcję wskaźnika pO_2/FiO_2 w okresie pierwszych 7 dni hospitalizacji w ramach dostępnej szerokości okna czasowego (*window*). Okno czasowe obejmuje 3 dni i podlega przesuwaniu w okresie 7 dni hospitalizacji. W oknie obserwowany jest przebieg wskaźnika pO_2/FiO_2. Na wejście AIS podawany był przypadek ze zbioru testowego, dla którego wykonano pomiary w podoknie czasowym danych wejściowych (*subwindow input*). Na wyjściu AIS była obserwowana odpowiedź, czyli wartości przewidywane, wskaźnika w podoknie czasowym predykcji (*subwindow output*). Odpowiedź sieci polega na przewidywaniu wartości wskaźnika pO_2/FiO_2 na okres wyznaczony przez:

$$subwindow\,output = window - subwindow\,input.$$

Na rysunku 7.1 został przedstawiony proces predykcji wskaźnika pO_2/FiO_2.



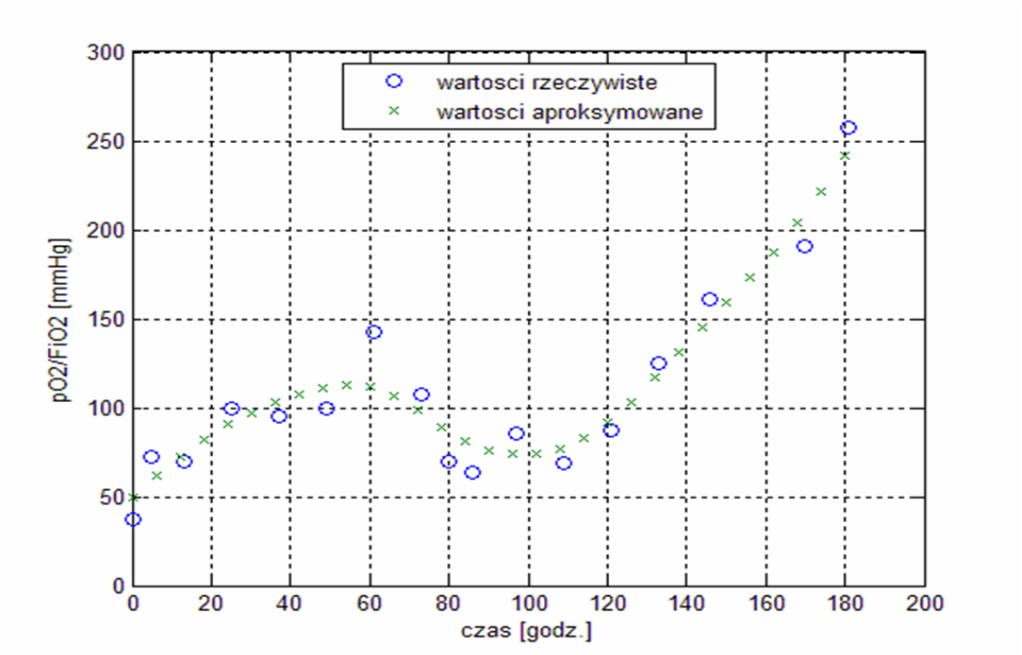
Rysunek 7.1: Proces predykcji wskaźnika pO_2/FiO_2

7.1.1 Badana grupa pacjentów

Badaną grupę pacjentów stanowili pacjenci z grup od 2 do 4 opisaną w rozdziale 2. Parametrami wejściowymi AIS są parametry: pO_2/FiO_2, pH, pCO_2, HCO_3. Parametrami wyjściowymi AIS były parametry takie same jak na wejściu AIS ale badanym parametrem wyjściowym był wskaźnik pO_2/FiO_2.

7.1.2 Przygotowanie danych

Pomiary gazometrii krwi: pO_2/FiO_2, pH, pCO_2, HCO_3 oraz nastawy respiratora FiO_2 nie są wykonywane w jednakowych odstępach czasu, co stwarza pewne trudności w trakcie analizy danych. Na tym etapie badań pod uwagę brane są jedynie te przypadki, które w przedziale 0-168 godz. miały więcej niż 4 pomiary parametrów gazometrii, a odstęp pomiędzy pomiarami nie przekraczał 50 godzin. Dane wejściowe do AIS powinny być podawane ze stałym krokiem przyrostu czasu. Problem ten rozwiązano przez zastosowanie aproksymacji. Przykład aproksymacji przedstawiono na rysunku 7.2. W wyniku zastosowania algorytmu aproksymacji uzyskano wartości ciągłe parametrów wejściowych. To pozwala na podawanie na wejście AIS wartości z równym krokiem przyrostu czasu. Wartość kroku czasowego przyjęto jako 6 godzin.



Rysunek 7.2: Proces aproksymacji wskaźnika pO_2/FiO_2

7.1.3 Predykcja wskaźnika pO_2/FiO_2 przez pierwsze trzy dni hospitalizacji pacjenta

Predykcja została przeprowadzona dla trzech pierwszych dni hospitalizacji pacjenta. Badanie polegało na podawaniu na wejście sieci immunologicznej przypadków ze zbioru testowego i obserwowanie odpowiedzi sztucznej sieci immunologicznej AIS. Badania przeprowadzono dla różnych podokien czasowych danych wejściowych (subwindows input) i podokien czasowych predykcji (subwindows output). W procesie uczenia parametry dla sieci immunologicznej miały wartości określone poniżej.

Dla sztucznej sieci immunologicznej AIS1 zadano wartości:

$$n = 8,\ \xi = 30\%,\ \sigma_d = 0.5,\ \sigma_s = 0.005.$$

Stopień dopasowania (powinowactwa) obliczany jest jako odległość skalarna zgodnie ze wzorem 3.5.

Dla sztucznej sieci immunologicznej AIS2 zadano wartości:

$$n = 4,\ \xi = 30\%,\ \sigma_d = 2.5,\ \sigma_s = 0.001.$$

Stopień dopasowania (powinowactwa) obliczany jest jako odległość euklidesowa zgodnie ze wzorem 3.2.

Przyjęto oznaczenia:

n liczba przeciwciał poddanych procesowi klonowania

ξ parametr określający, jaki procent populacji przeciwciał będzie poddawane procesowi klonowania w procesie aktywacji antygenem

σ_d próg usuwania komórek

σ_s próg supresji.

Kryteria stopu dla algorytmu uczenia są związane z liczbą generacji i w tym przypadku liczba ta wynosiła $Ngen = 100$.

Podczas uczenia i testu zostały użyte następujące zbiory:

Zbiór uczący (132/4/72), czyli

- 132 pacjentów
- 4 parametry wejściowe: pO_2/FiO_2, pH, pCO_2, HCO_3
- okno czasowe (window): 72 godz. (3 dni)

Zbiór testowy (32/4/x), czyli

- 32 pacjentów
- 4 parametry wyjściowe: pO_2/FiO_2, pH, pCO_2, HCO_3
- podokno czasowe danych wejściowych (subwindow input):

$x = 6$, 12, 18, 24, 30, 36, 42, 48 [godz.].

Wyniki predykcji wskaźnika pO_2/FiO_2 opisano w tablicy 7.1.

7.2 Predykcja wskaźnika pO_2/FiO_2 przy użyciu modelu postępowania lekarza (MPL)

Opis modelu postępowania lekrza do predykcji wskaźnika pO_2/FiO_2 znajduje się w rozdziale 5. Predykcja wskaźnika pO_2/FiO_2 przez zastosowanie MPL polegała na podawaniu na wejście przypadków ze zbioru testowego i obserwowanie dopasowanych z bazy danych przypadków, u których przebieg powikłań oddechowych był zbliżony do rozważanego przypadku wejściowego. Dopasowanie oparte jest o oryginalne dane pacjentów. MPL umożliwia predykcję w okresie pierwszych siedmiu dni hospitalizacji.

7.2.1 Badana grupa pacjentów

Badaną grupę pacjentów stanowili pacjenci ze zbioru testowego. Jest to ten sam zbiór testowy który został użyty do testu AIS. Pozwoliło to na porównanie wyników AIS i MPL.

Parametrami wejściowymi i wyjściowymi MPL były parametry: pO_2/FiO_2, pH, pCO_2, HCO_3, wiek płodowy (ang. gestational age), zastosowanie surfaktantu (ang. medications - surfactant), fakt wystąpienia chorób: odma, posocznica, NEC, PDA, IVH, wystąpienie zgonu do końca 2 tygodnia życia. Parametrami wyjściowymi MPL były parametry takie same jak na wejściu MPL, ale badanym parametrem wyjściowym był wskaźnik pO_2/FiO_2.

7.2.2 Predykcja wskaźnika pO_2/FiO_2 dla pierwszych 3 dni hospitalizacji pacjenta

Predykcja została przeprowadzona dla trzech pierwszych dni hospitalizacji pacjenta. Badania przeprowadzono dla tych samych podokien czasowych predykcji (subwindows output) jak przy metodzie AIS. Pozwoliło to na porównanie wyników z wynikami metody AIS. Wyniki predykcji wskaźnika pO_2/FiO_2 znajdują się w tablicy 7.1.

7.3 Błędy metod

Błąd predykcji, dla pacjenta testowego, dla każdej godziny predykcji obliczany był według wzoru:

$$e_i = (|x_i - y_i|/x_i) \cdot 100 \tag{7.1}$$

gdzie:

x_i rzeczywisty przebieg wskaźnika pO_2/FiO_2 pacjenta testowego,

y_i wyjściowy przebieg wskaźnika pO_2/FiO_2 uzyskany z odpowiedzi AIS lub MPL,

i godzina predykcji.

Błąd predykcji, dla zbioru testowego, dla każdej godziny predykcji obliczany był jako błąd średni dla wszystkich pacjentów testowych:

$$et_i = \frac{1}{n} \sum_{k=1}^{k=n} \cdot e_i \tag{7.2}$$

gdzie:

n liczba pacjentów testowych,

i godzina predykcji.

7.4 Wyniki dla AIS i MPL

W tablicy 7.1 przedstawiono wyniki predykcji pO_2/FiO_2 dla zbioru testowego w poszczególnych podoknach czasowych (subwindows). Do predykcji zastosowano metody:

- sztuczną sieć immunologiczną AIS1
- sztuczną sieć immunologiczną AIS2
- model postępowania lekarza MPL.

Metody	subwindow input	subwindow output	Błąd metody dla godzin:										
			12	18	24	30	36	42	48	54	60	66	72
	godz.	godz.	%	%	%	%	%	%	%	%	%	%	%
AIS1	6	66	17,1	21,4	26,3	29,8	30,0	26,7	19,7	12,9	8,9	10,6	15,2
AIS2	6	66	12,7	21,0	26,2	28,8	26,7	27,5	19,7	15,8	15,8	13,2	15,5
MPL	6	66	6,7	9,7	13,6	18,4	24,5	29,9	34,9	38,5	40,7	42,1	44,6
AIS1	12	60		12,4	16,2	19,5	20,6	19,8	16,9	14,5	13,4	12,5	15,9
AIS2	12	60		15,6	19,6	20,5	23,2	25,5	24,7	21,1	16,6	10,0	8,4
MPL	12	60		11,7	16,9	24,0	32,1	39,5	45,2	47,6	47,6	45,6	47,0
AIS1	18	54			13,3	16,6	16,7	16,8	15,4	14,4	13,4	10,9	10,2
AIS2	18	54			10,4	12,5	16,5	22,8	26,9	20,7	16,1	9,5	8,1
MPL	18	54			7,9	10,0	14,2	19,6	24,2	27,9	31,6	35,0	38,7
AIS1	24	48				14,2	15,4	15,6	14,9	14,6	13,1	11,6	10,8
AIS2	24	48				13,7	15,3	20,1	23,0	19,0	14,2	8,5	8,8
MPL	24	48				2,5	5,0	9,0	12,2	15,6	21,1	26,8	32,9
AIS1	30	42					9,8	10,1	9,8	10,3	9,6	9,4	9,6
AIS2	30	42					13,6	19,6	22,7	18,0	15,8	9,4	8,4
MPL	30	42					5,0	9,0	12,2	15,6	21,1	26,8	32,9
AIS1	36	36						9,2	8,9	8,9	8,7	9,5	10,7
AIS2	36	36						18,0	20,7	17,4	14,7	10,0	7,6
MPL	36	36						7,9	11,2	15,5	23,4	32,6	43,3
AIS1	42	30							11,2	10,5	9,2	8,4	8,9
AIS2	42	30							13,8	10,9	10,8	9,0	8,9
MPL	42	30							5,2	7,7	16,6	28,4	43,4
AIS1	48	24								8,4	7,7	8,1	8,7
AIS2	48	24								10,0	10,3	8,4	8,6
MPL	48	24								7,7	16,6	28,4	43,4

Tablica 7.1: Wyniki predykcji wskaźnika pO_2/FiO_2

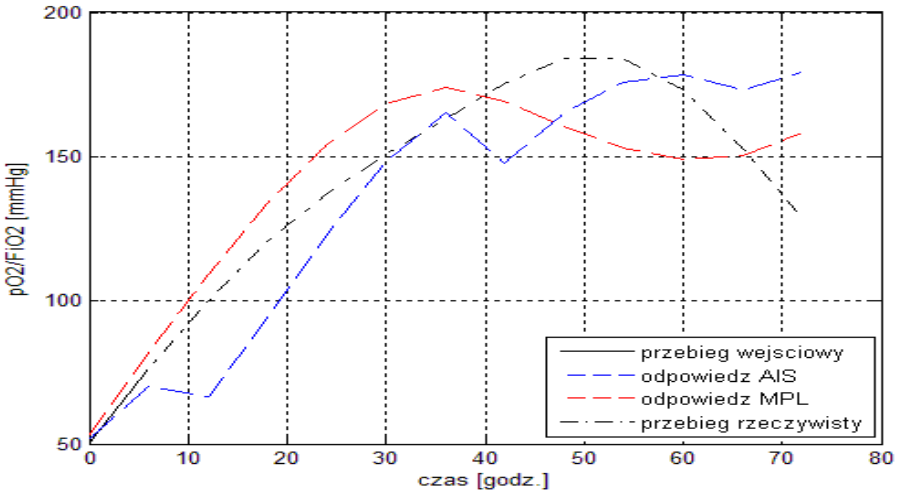
7.4.1 Wykresy predykcji wskaźnika pO_2/FiO_2

Poniżej przedstawiono wykresy predykcji i błędu predykcji wskaźnika pO_2/FiO_2 dla wybranych przypadków ze zbioru testowego.

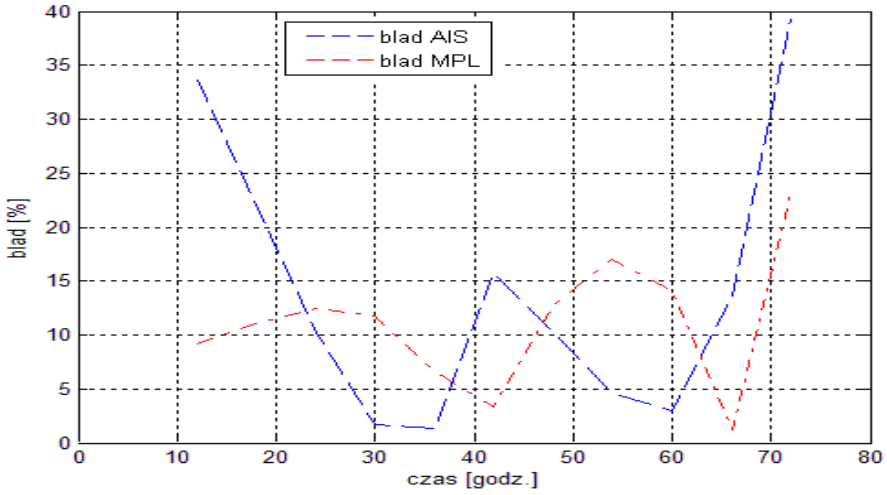
Okno czasowe (window) wybrane do obliczania stanów pacjenta zostało określone na 72 godziny.

Odpowiednie podokna czasowe (subwindows) zostały określone odpowiednio dla danych wejściowych (subwindows input) jako 6, 24 oraz 48 godzin. Natomiast podokna czasowe dla wartości wyjściowych (subwindows output), predykowanych za pomocą różnych metod, zostały określone odpowiednio jako 66, 48 oraz 24 godziny.

Podokno danych wejściowych (subwindow input) wynosi 6 godzin.
Podokno predykcji (subwindows output) wynosi 66 godzin.
Zastosowane metody AIS1 i MPL.
- uzyskany rezultat charakteryzuje się najmniejszym błędem metody

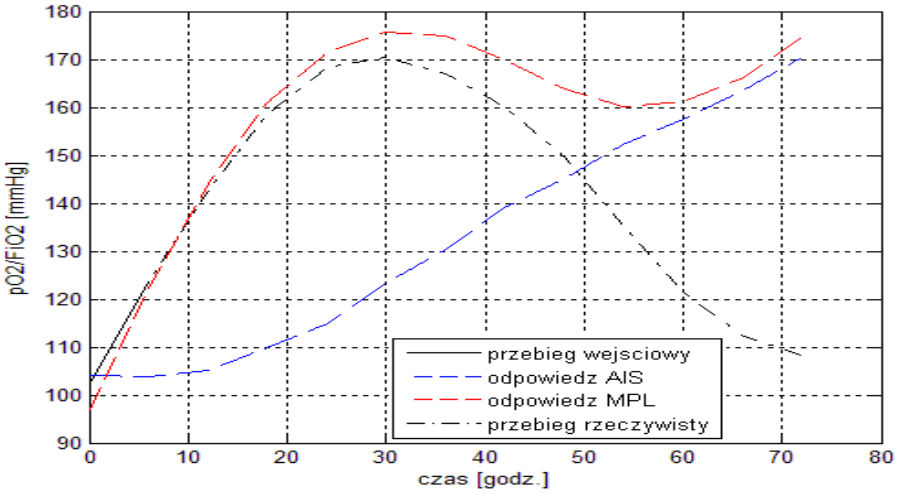


Rysunek 7.3: Wykres predykcji wskaźnika pO_2/FiO_2 - AIS1-MPL: 12_72

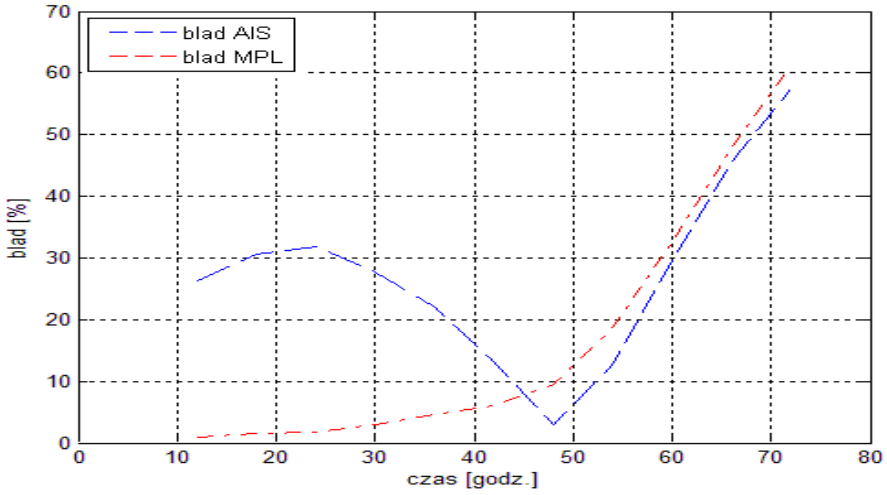


Rysunek 7.4: Wykres błędu predykcji wskaźnika pO_2/FiO_2 - AIS1-MPL: 12_72

- rezultat obarczony największym błędem metody

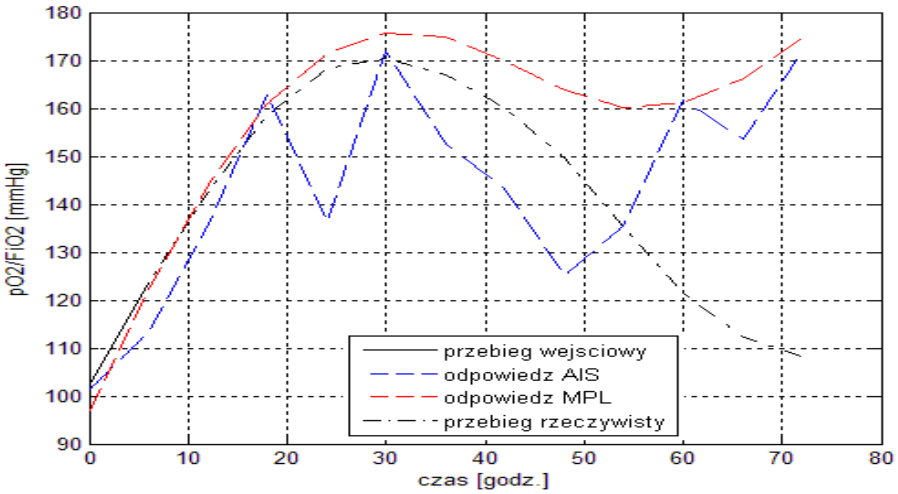


Rysunek 7.5: Wykres predykcji wskaźnika pO_2/FiO_2 - AIS1-MPL: 12_72

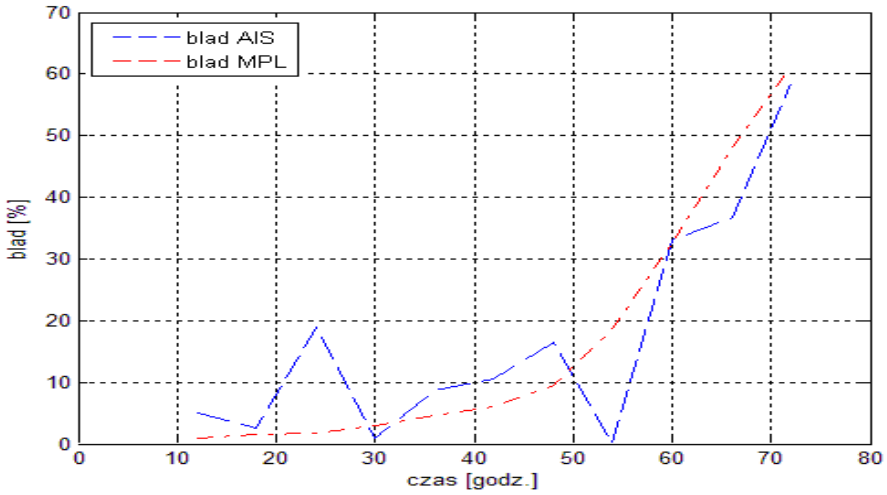


Rysunek 7.6: Wykres błędu predykcji wskaźnika pO_2/FiO_2 - AIS1-MPL: 12_72

Zastosowane metody AIS2 i MPL.
- uzyskany rezultat charakteryzuje się najmniejszym błędem metody

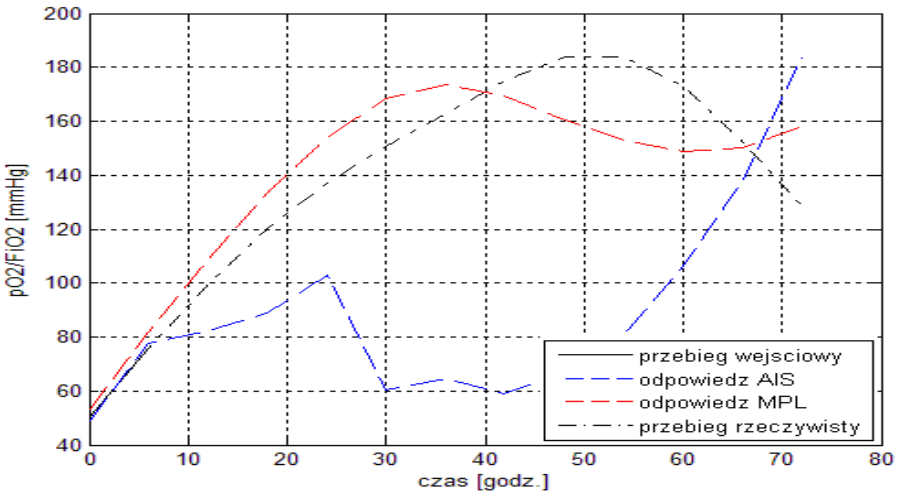


Rysunek 7.7: Wykres predykcji wskaźnika pO_2/FiO_2 - AIS2-MPL: 12_72

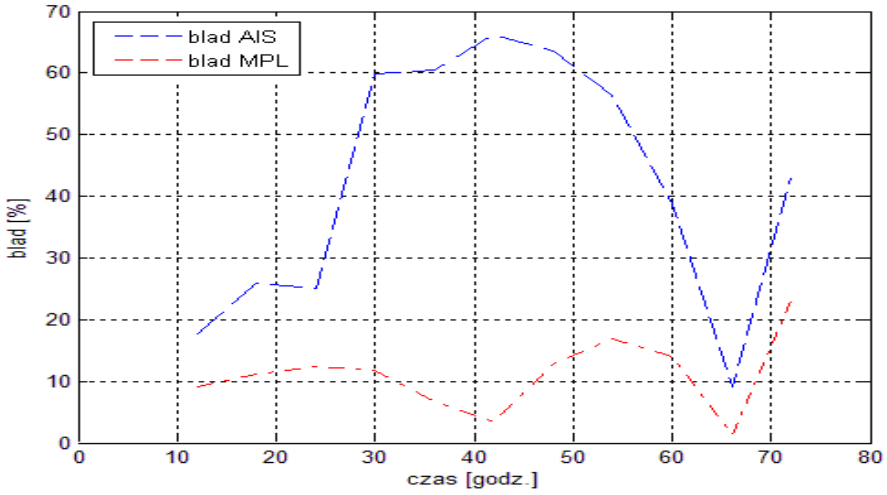


Rysunek 7.8: Wykres błędu predykcji wskaźnika pO_2/FiO_2 - AIS2-MPL: 12_72

- rezultat obarczony największym błędem metody

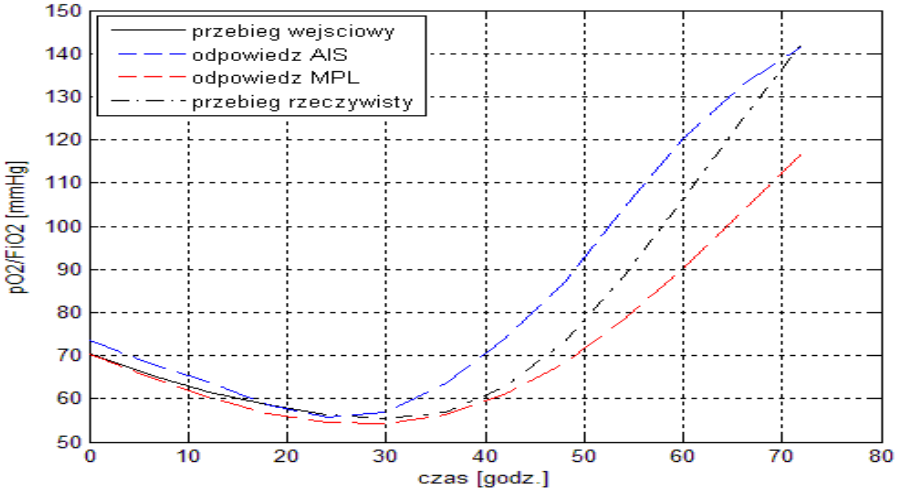


Rysunek 7.9: Wykres predykcji wskaźnika pO_2/FiO_2 - AIS2-MPL: 12_72

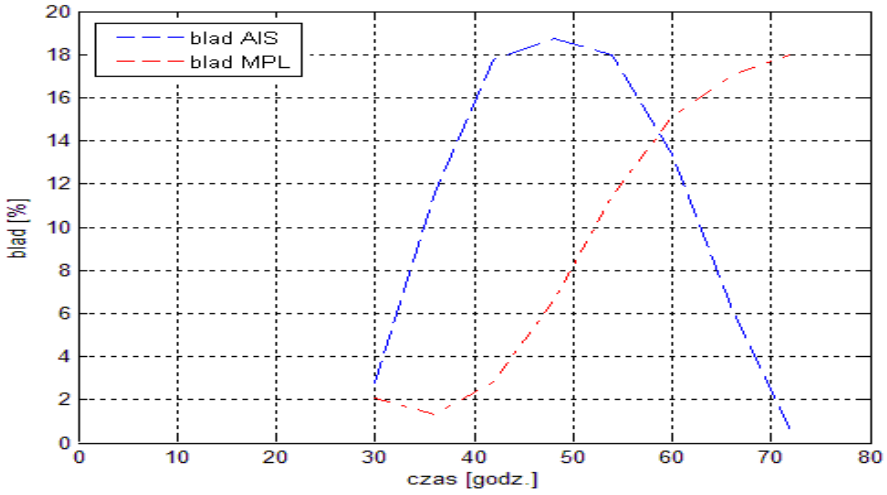


Rysunek 7.10: Wykres błędu predykcji wskaźnika pO_2/FiO_2 - AIS2-MPL: 12_72

Podokno danych wejściowych (subwindow input) wynosi 24 godziny.
Podokno predykcji (subwindows output) wynosi 48 godzin.
Zastosowane metody AIS1 i MPL.
- uzyskany rezultat charakteryzuje się najmniejszym błędem metody

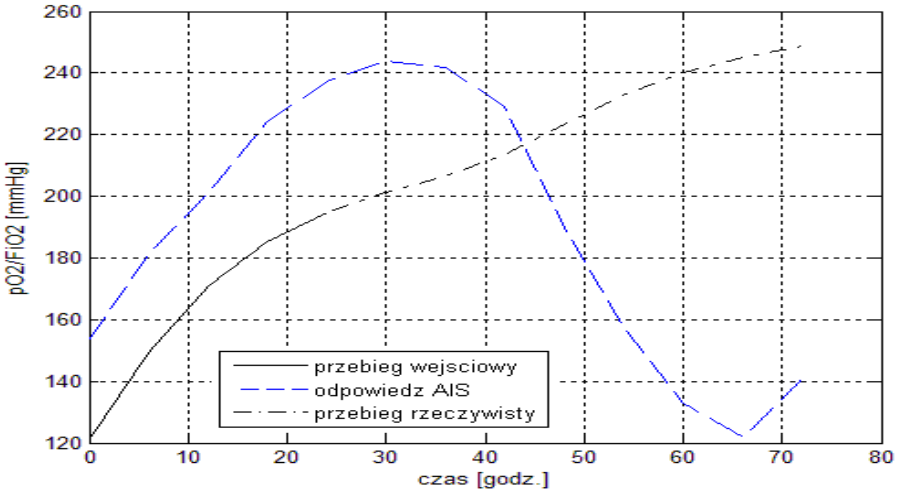


Rysunek 7.11: Wykres predykcji wskaźnika pO_2/FiO_2 - AIS1-MPL: 30_72

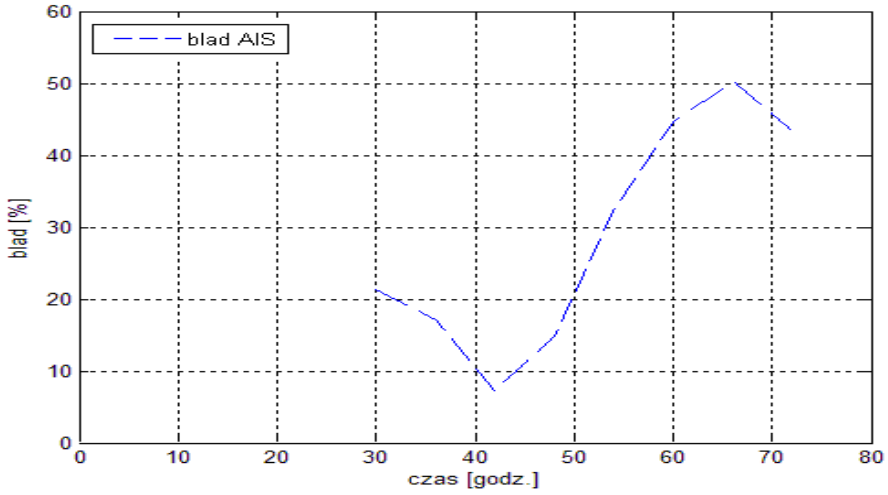


Rysunek 7.12: Wykres błędu predykcji wskaźnika pO_2/FiO_2 - AIS1-MPL: 30_72

- rezultat obarczony największym błędem metody

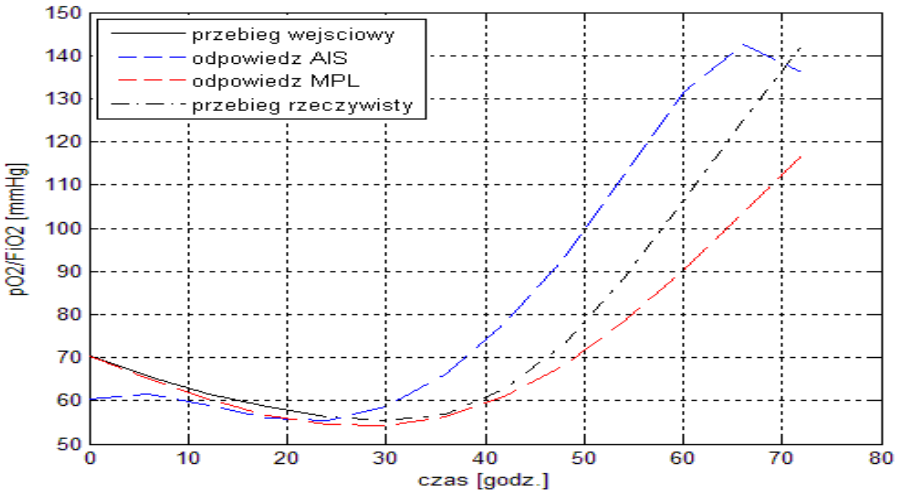


Rysunek 7.13: Wykres predykcji wskaźnika pO_2/FiO_2 - AIS1-MPL: 30_72

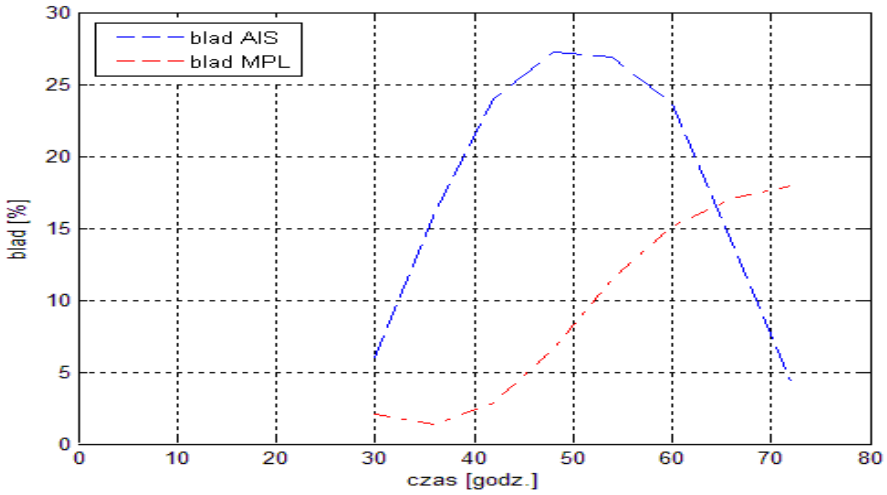


Rysunek 7.14: Wykres błędu predykcji wskaźnika pO_2/FiO_2 - AIS1-MPL: 30_72

Zastosowane metody AIS2 i MPL.
- uzyskany rezultat charakteryzuje się najmniejszym błędem metody

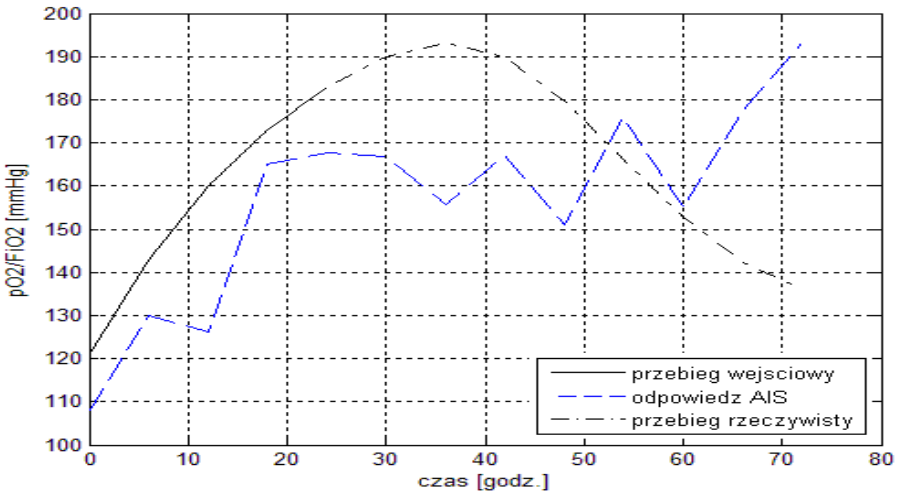


Rysunek 7.15: Wykres predykcji wskaźnika pO_2/FiO_2 - AIS2-MPL: 30_72

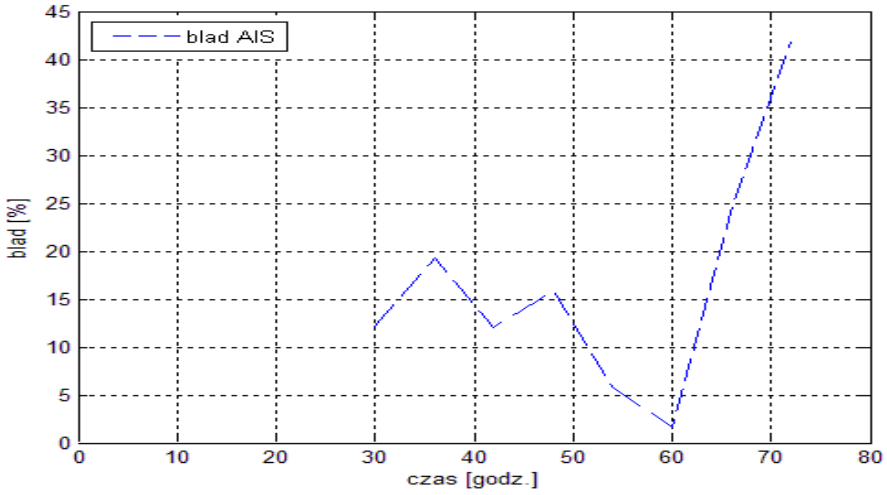


Rysunek 7.16: Wykres błędu predykcji wskaźnika pO_2/FiO_2 - AIS2-MPL: 30_72

- rezultat obarczony największym błędem metody

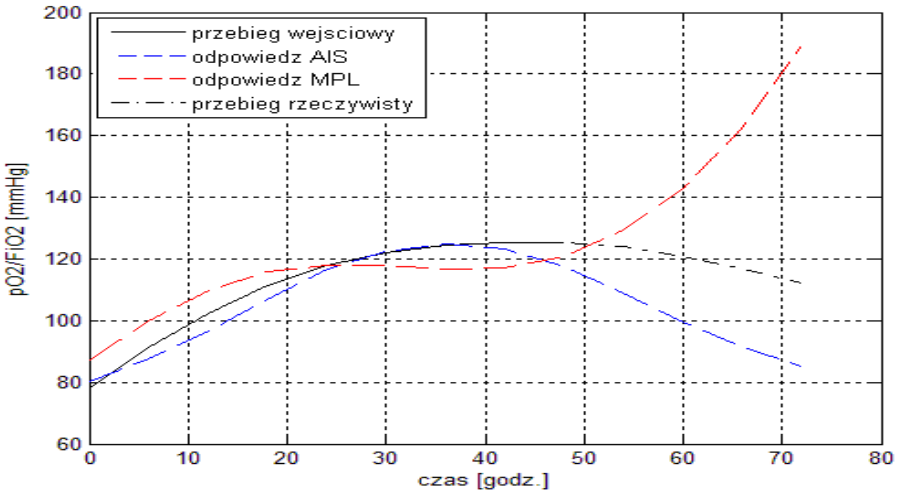


Rysunek 7.17: Wykres predykcji wskaźnika pO_2/FiO_2 - AIS2-MPL: 30_72

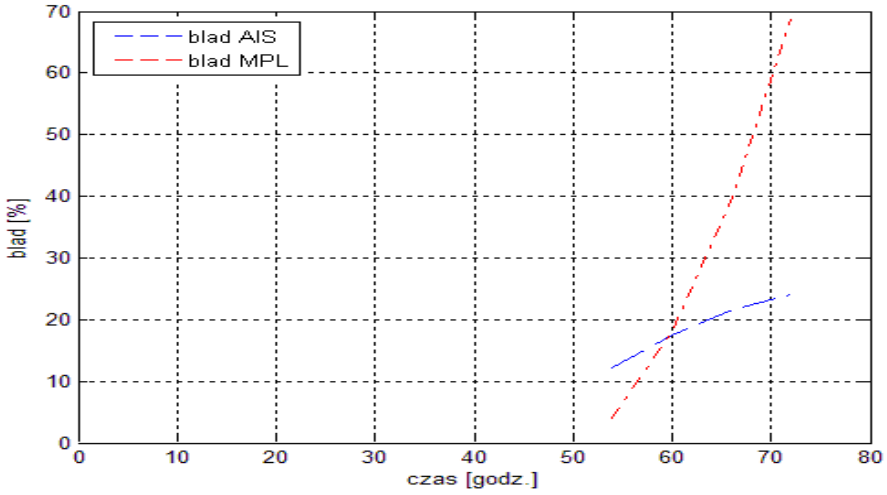


Rysunek 7.18: Wykres błędu predykcji wskaźnika pO_2/FiO_2 - AIS2-MPL: 30_72

Podokno danych wejściowych (subwindow input) wynosi 48 godzin.
Podokno predykcji (subwindows output) wynosi 24 godziny.
Zastosowane metody AIS1 i MPL.
- uzyskany rezultat charakteryzuje się najmniejszym błędem metody

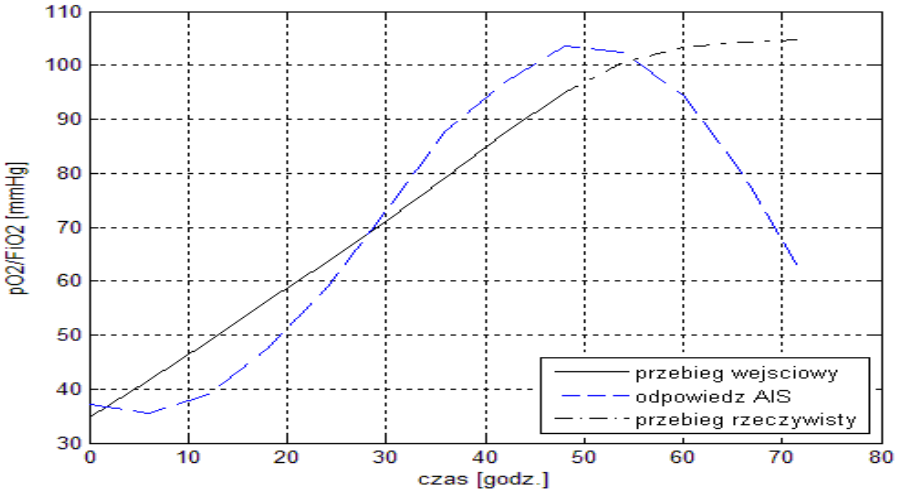


Rysunek 7.19: Wykres predykcji wskaźnika pO_2/FiO_2 - AIS1-MPL: 54_72

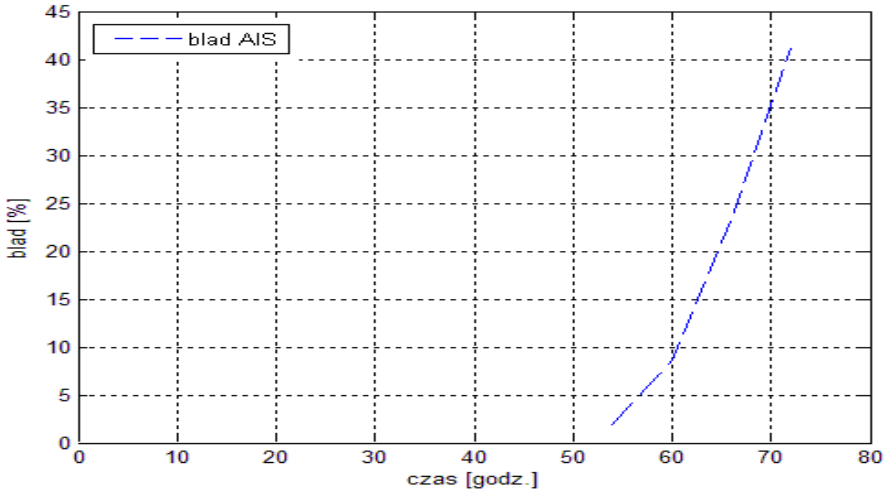


Rysunek 7.20: Wykres błędu predykcji wskaźnika pO_2/FiO_2 - AIS1-MPL: 54_72

- rezultat obarczony największym błędem metody

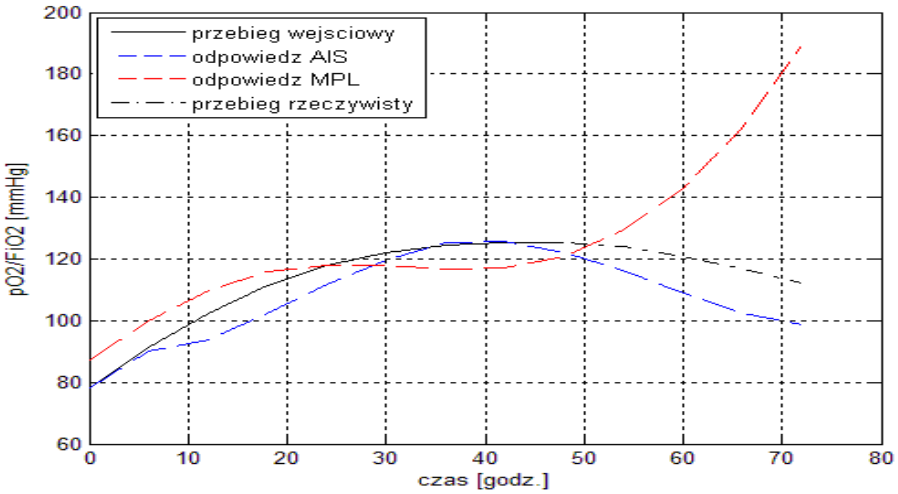


Rysunek 7.21: Wykres predykcji wskaźnika pO_2/FiO_2 - AIS1-MPL: 54_72

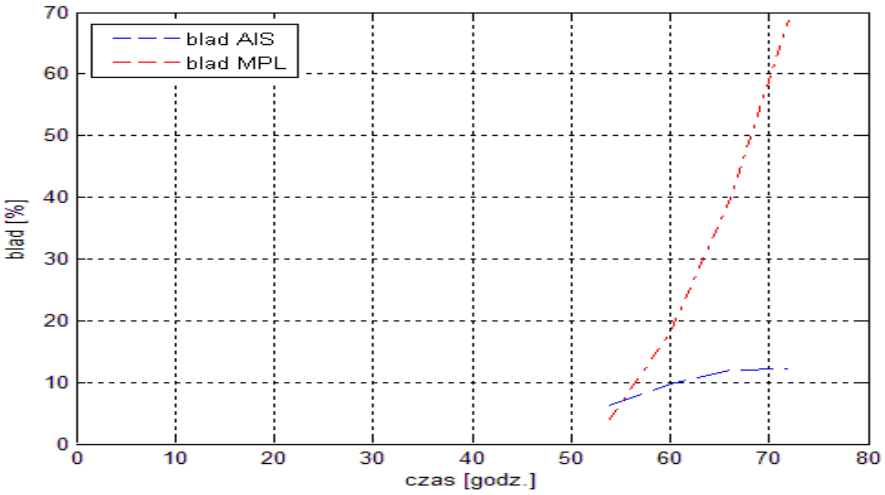


Rysunek 7.22: Wykres błędu predykcji wskaźnika pO_2/FiO_2 - AIS1-MPL: 54_72

Zastosowane metody AIS2 i MPL.
- uzyskany rezultat charakteryzuje się najmniejszym błędem metody

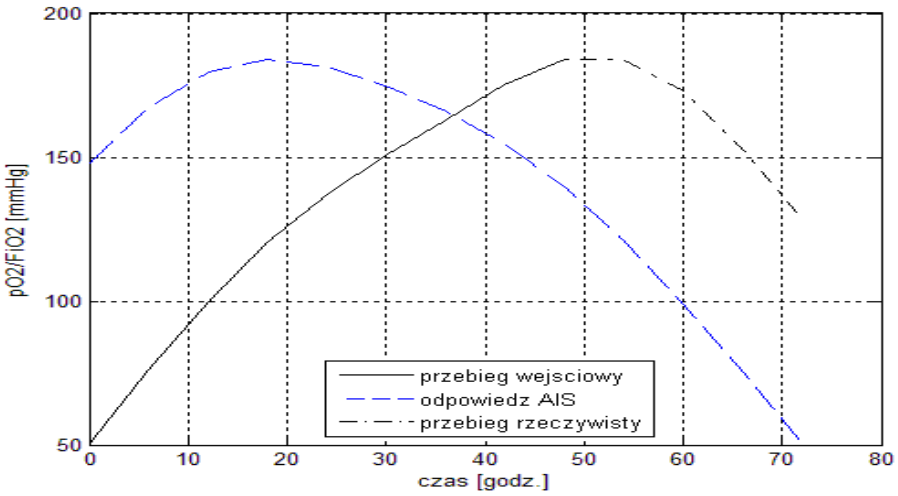


Rysunek 7.23: Wykres predykcji wskaźnika pO_2/FiO_2 - AIS2-MPL: 54_72

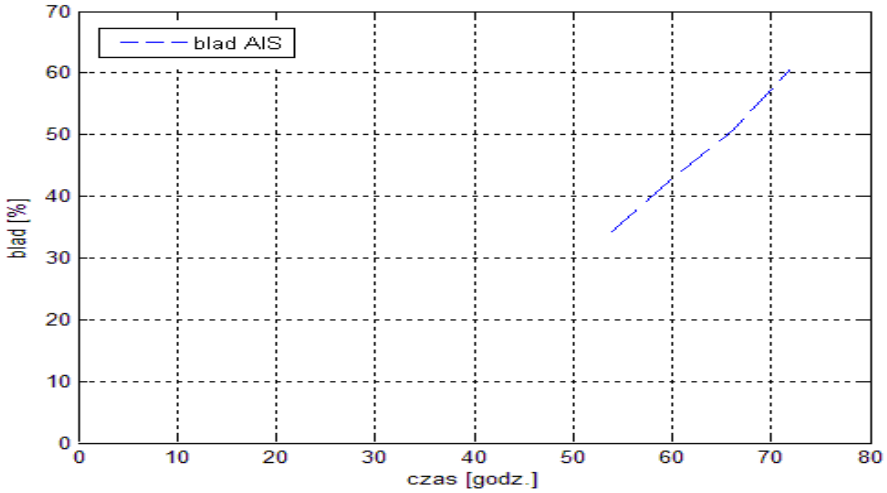


Rysunek 7.24: Wykres błędu predykcji wskaźnika pO_2/FiO_2 - AIS2-MPL: 54_72

- rezultat obarczony największym błędem metody



Rysunek 7.25: Wykres predykcji wskaźnika pO_2/FiO_2 - AIS2-MPL: 54_72

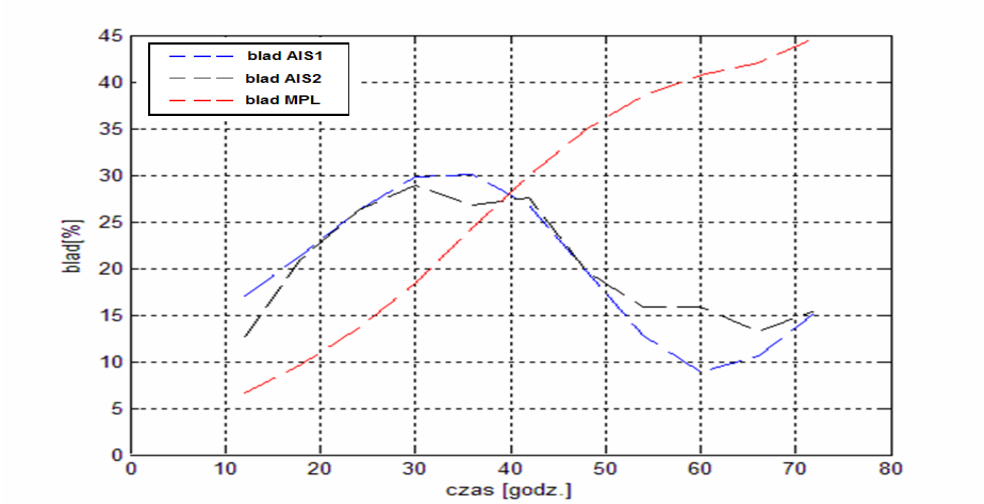


Rysunek 7.26: Wykres błędu predykcji wskaźnika pO_2/FiO_2 - AIS2-MPL: 54_72

7.5 Porównanie wyników metody AIS z metodą MPL

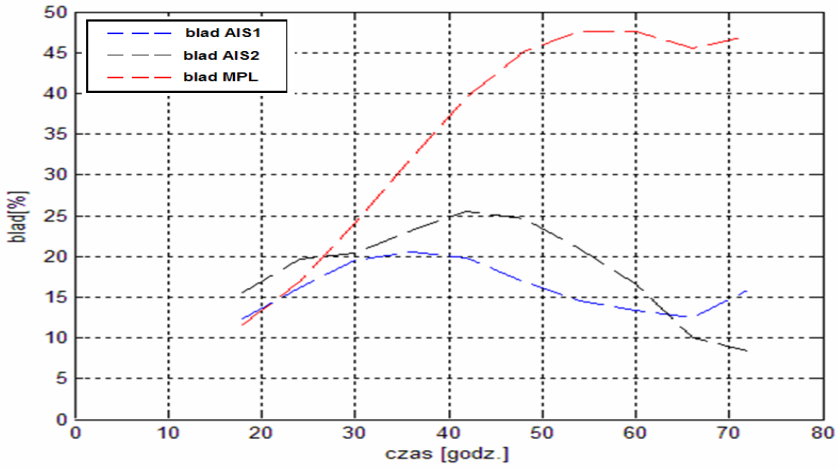
Na wykresach zostały przedstawione błędy predykcji AIS i MPL dla wszystkich podokien czasowych (subwindows).

Podokno danych wejściowych (subwindow input) wynosi 6 godzin.
Podokno predykcji (subwindows output) wynosi 66 godzin.



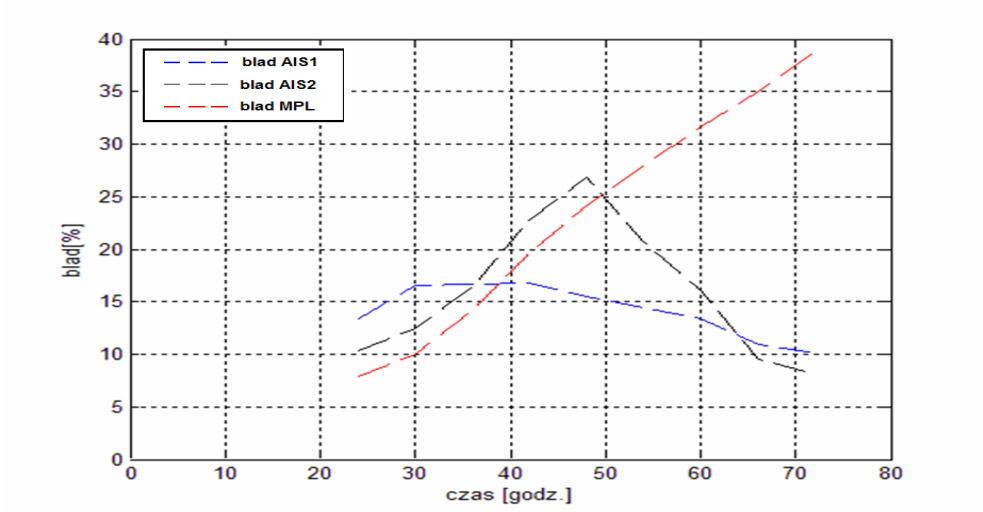
Rysunek 7.27: Wykres błędu predykcji wskaźnika pO_2/FiO_2 : 12_72

Podokno danych wejściowych (subwindow input) wynosi 12 godzin.
Podokno predykcji (subwindows output) wynosi 60 godzin.



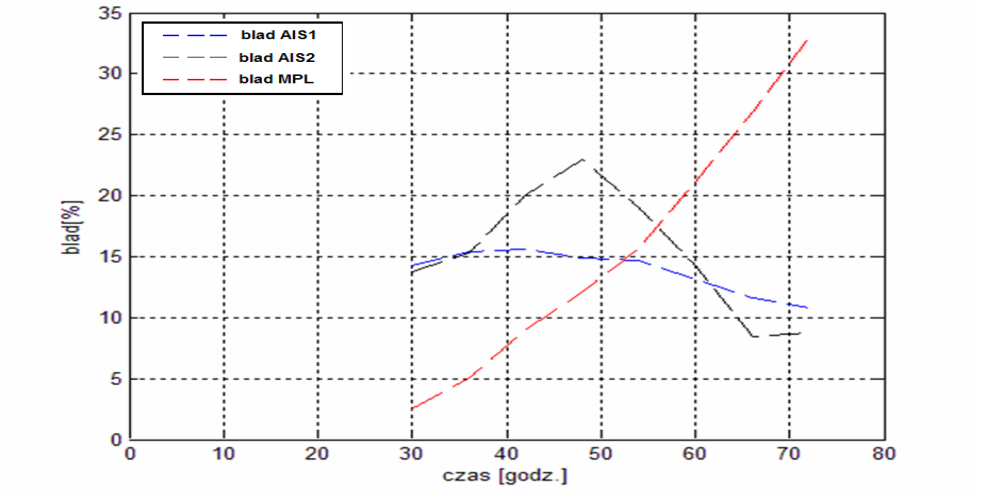
Rysunek 7.28: Wykres błędu predykcji wskaźnika pO_2/FiO_2 : 18_72

Podokno danych wejściowych (subwindow input) wynosi 18 godzin. Podokno predykcji (subwindows output) wynosi 54 godziny.



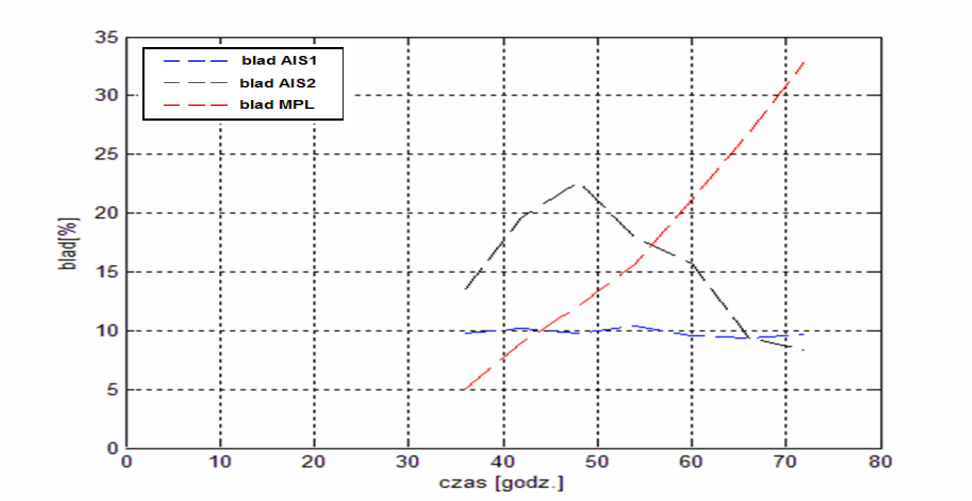
Rysunek 7.29: Wykres błędu predykcji wskaźnika pO_2/FiO_2 : 24_72

Podokno danych wejściowych (subwindow input) wynosi 24 godziny. Podokno predykcji (subwindows output) wynosi 48 godzin.



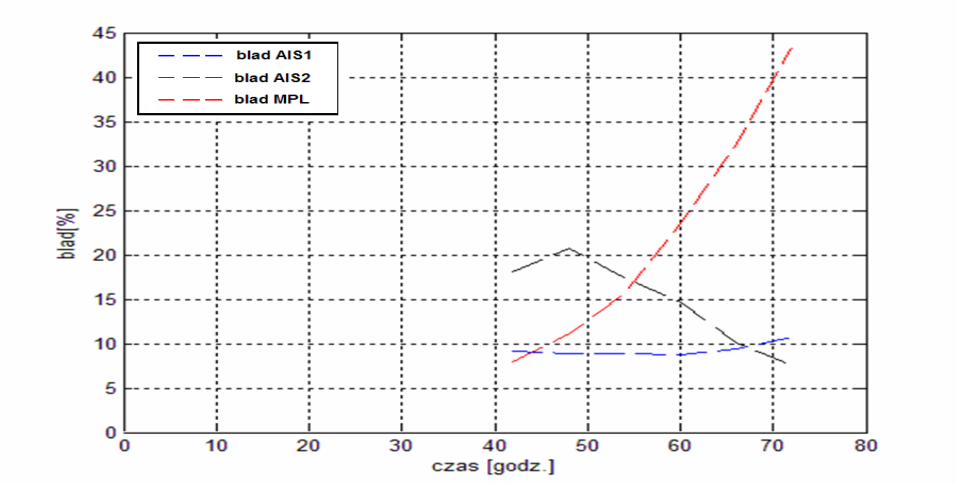
Rysunek 7.30: Wykres błędu predykcji wskaźnika pO_2/FiO_2 : 30_72

Podokno danych wejściowych (subwindow input) wynosi 30 godzin. Podokno predykcji (subwindows output) wynosi 42 godziny.



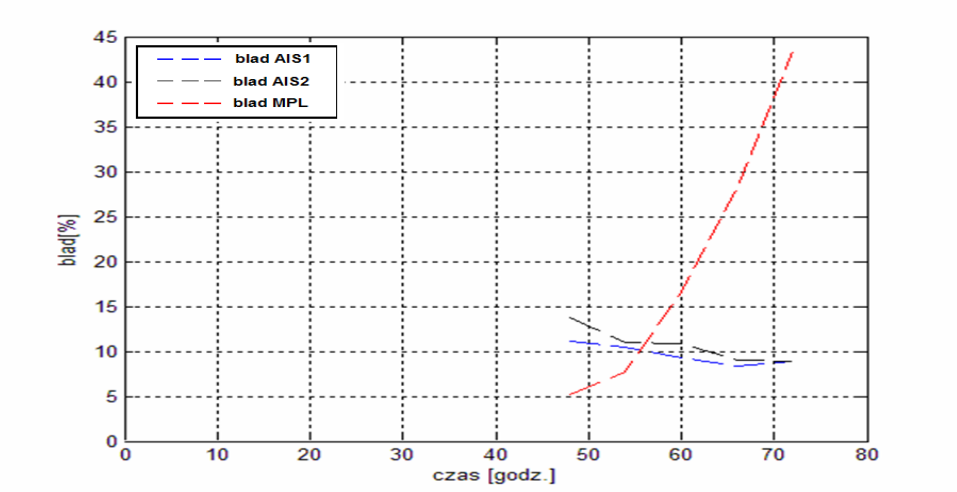
Rysunek 7.31: Wykres błędu predykcji wskaźnika pO_2/FiO_2 : 36_72

Podokno danych wejściowych (subwindow input) wynosi 36 godzin. Podokno predykcji (subwindows output) wynosi 36 godzin.



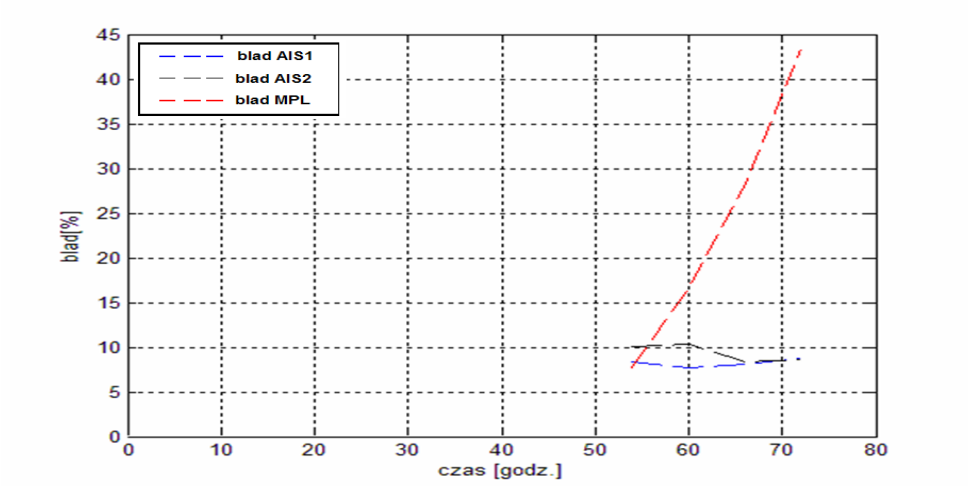
Rysunek 7.32: Wykres błędu predykcji wskaźnika pO_2/FiO_2 : 40_72

Podokno danych wejściowych (subwindow input) wynosi 42 godziny. Podokno predykcji (subwindows output) wynosi 30 godzin.



Rysunek 7.33: Wykres błędu predykcji wskaźnika pO_2/FiO_2 : 48_72

Podokno danych wejściowych (subwindow input) wynosi 48 godzin. Podokno predykcji (subwindows output) wynosi 24 godziny.



Rysunek 7.34: Wykres błędu predykcji wskaźnika pO_2/FiO_2 : 54_72

7.6 Predykcja wskaźnika pO_2/FiO_2 dla 18 godziny metodą AIS z zastosowaniem algorytmu nr 1 i algorytmu nr 2

Predykcję przeprowadzono zgodnie z algorytmem nr 1 i nr 2 opisanymi w rozdziale 3. Predykcja wskaźnika pO_2/FiO_2 dla 18 godziny polegała na podawaniu na wejście sieci przypadków ze zbioru testowego.

Parametrami wejściowymi są wartości pO_2/FiO_2 i pH dla 0, 6 i 12 godziny. Parametrem wyjściowym sieci jest wartość pO_2/FiO_2 dla 18 godziny.

Podczas uczenia i testu zostały użyte następujące zbiory:

Zbiór uczący obejmował 120 pacjentów, 2 parametry wejściowe: pO_2/FiO_2, pH dla 0, 6, 12 i 18 godziny.

Zbiór testowy zamieszczono w tablicy 7.2. Obejmował on 8 pacjentów,

2 parametry wejściowe: pO_2/FiO_2, pH dla 0, 6 i 12 godziny.

	pO_2/FiO_2				pH			
Pacjent	0 godz.	6 godz.	12 godz.	18 godz.	0 godz.	6 godz.	12 godz.	18 godz.
1	93,13	103,86	116,13	131,51	7,39	7,40	7,41	7,41
2	109,43	128,74	149,43	169,87	7,47	7,46	7,45	7,44
3	68,95	73,45	78,96	85,86	7,20	7,21	7,21	7,23
4	121,34	142,74	160,11	173,01	7,35	7,37	7,38	7,39
5	74,13	81,04	87,72	92,86	7,32	7,34	7,36	7,38
6	64,35	63,78	61,47	58,18	7,28	7,28	7,28	7,28
7	113,69	138,14	160,64	176,53	7,32	7,31	7,30	7,29
8	133,80	121,46	118,11	110,16	7,35	7,34	7,33	7,31

Tablica 7.2: Zbiór testowy

Wyniki predykcji wskaźnika pO_2/FiO_2 dla 18 godziny (AIS algorytm nr 1) przedstawia tablica 7.3.

	pO_2/FiO_2			
Pacjent	0 godz.	6 godz.	12 godz.	18 godz.
1	93,09	106,07	120,10	134,33
2	109,38	129,10	150,09	167,51
3	69,15	73,79	79,33	84,78
4	121,34	142,52	160,32	175,77
5	75,08	82,37	88,64	92,91
6	64,35	63,27	62,26	61,57
7	113,68	138,88	160,49	176,20
8	133,43	120,42	116,13	105,34

Tablica 7.3: Wyniki predykcji pO_2/FiO_2 dla 18 godziny - AIS algorytm nr 1

Wyniki predykcji wskaźnika pO_2/FiO_2 dla 18 godziny (AIS algorytm nr 2) przedstawia tablica 7.4.

	pO_2/FiO_2			
Pacjent	0 godz.	6 godz.	12 godz.	18 godz.
1	90,55	102,32	112,69	122,35
2	109,63	122,61	138,06	154,30
3	65,52	73,00	79,73	84,89
4	117,65	135,75	153,24	169,51
5	74,12	82,96	91,01	98,25
6	67,86	63,77	61,40	61,83
7	117,65	135,75	153,24	169,51
8	135,86	123,95	114,94	107,97

Tablica 7.4: Wyniki predykcji pO_2/FiO_2 dla 18 godziny - AIS algorytm nr 2

Tablica 7.5 przedstawia błędy predykcji pO_2/FiO_2 dla 18 godziny - AIS nr 1 i AIS nr 2.

	Błąd	
	AIS nr 1	AIS nr 2
Pacjent	[%]	[%]
1	2,1	7,0
2	1,4	9,2
3	1,3	1,1
4	1,6	2,0
5	0,1	5,8
6	5,8	6,3
7	0,2	4,0
8	4,4	2,0

Tablica 7.5: Błędy predykcji pO_2/FiO_2 dla 18 godziny - AIS nr 1 i AIS nr 2

7.7 Wnioski

Predykcję wskaźnika pO_2/FiO_2 przeprowadzono w okresie trzech pierwszych dni hospitalizacji pacjenta. Wyniki znajdują się w tablicy 7.1.

Zdolności predykcyjne AIS zależą zarówno od parametrów ξ, σ_d, σ_s jak i od sposobu obliczania odległości (powinowactwa).

Badania AIS przeprowadzono dla dwóch odległości:

- euklidesowa, wzór 3.2 - (AIS1)
- iloczyn skalarny, wzór 3.5 - (AIS2).

Mniejszy błąd predykcji uzyskano przy zastosowaniu odległości skalarnej (AIS1).

Okres predykcji wynosił: 66, 60, 54, 48, 42, 36, 30, 24 [godz.] Zdolność predykcyjna sztucznej sieci immunologicznej wzrasta wraz ze wzrostem podokna danych wejściowych (subwindow input), a tym samym zmniejszeniem podokna predykcji (subwindow output). Zdolność predykcyjna AIS wzrasta wraz ze zmniejszeniem okresu predykcji.

Na przykład błąd predykcji dla 72 godziny wynosił 15,2 % dla podokna danych wejściowych (subwindow input) z obserwacji 6 godzinnych a 8,7 % dla podokna danych wejściowych (subwindow input) z obserwacji 48 godzinnych.

Nawet wtedy gdy bezwzględne wartości błędu prognozy przekraczały maksymalny dopuszczalny błąd właściwie prognozowany był trend zmian wskaźnika pO_2/FiO_2 .

Zdolność predykcji AIS porównano z predykcją modelu postępowania lekarza MPL. Dla AIS i MPL zastosowano ten sam zbiór testowy.

W początkowym okresie podokna predykcji (subwindow input) błąd predykcji MPL jest mniejszy od AIS, lecz w późniejszym okresie znacznie wzrasta i jest większy od błędu predykcji AIS.

Na przykład błąd predykcji dla 12 godziny metodą AIS wynosił 12,7 % a dla metody MPL wynosił 6,7 %.

Na przykład błąd predykcji dla 72 godziny metodą AIS wynosił 15,2 % a dla metody MPL wynosił 44,6 %.

W niektórych przypadkach model MPL nie znalazł dopasowania w bazie danych dla rozpatrywanego przypadku. Są to przypadki zamieszczone na rysunkach: 7.13, 7.17, 7.21 oraz 7.25.

Rozdział 8

Wyniki SVM i SSN - ocena ryzyka zgonu pacjenta

Proces uczenia i testowania Support Vector Machines (SVM) był realizowany w środowisku obliczeniowym Python 2.5.1. Funkcje użyte do uczenia i klasyfikacji zostały opisane w dodatku A.

Proces uczenia i testowania Sztucznych Sieci Neuronowych (SSN) był realizowany w środowisku obliczeniowym MATLAB 7.1 (R14). Funkcje użyte do uczenia i testowania zostały opisane w dodatku A.

8.1 Prognoza zgonu pacjenta – metoda SVM i SSN

Zadaniem metody SVM i SSN jest prognoza zgonu pacjenta (w ciągu dwóch pierwszych tygodni hospitalizacji) dla każdego z pierwszych 3 dni hospitalizacji. Prognoza jest przedstawiona jako: wysokie ryzyko zgonu – TAK/NIE.

8.1.1 Badana grupa pacjentów

Badaną grupę pacjentów stanowili pacjenci, którzy w badanym dniu hospitalizacji należeli do grupy 4 [rozdz.2].

Badaną grupą pacjentów była grupa dzieci urodzonych przedwcześnie z masą ciała $<$ 1500g w Oddziale Intensywnej Terapii Noworodka, przyjętych na oddział do II doby życia, ze skrajnymi zaburzeniami oddechowymi (wysokie ryzyko zgonu z przyczyn oddechowych) pO2 / FiO2 $<$ 100 mmHg i wsparcie oddechowe – IMV, SIMV.

8.1.2 Parametry wejściowe

Parametrami wejściowymi są:

- pO_2
- FiO_2
- pO_2/FiO_2

- pH
- pCO_2
- HCO_3
- BE
- wiek płodowy (gestational age)
- zastosowanie surfaktantu (medications – surfactant)
- fakt wystąpienia chorób: odma, posocznica, NEC, PDA, IVH
- płeć
- wystąpienie zgonu do końca 2 tygodnia życia.

Parametry wejściowe zostały podzielone na 5 zestawów danych:

Zestaw 1	Zestaw 2	Zestaw 3	Zestaw 4	Zestaw 5
zgon do 2 tyg. życia	zgon do 2 tyg. życia	zgon do 2 tyg. życia	zgon do 2 tyg. życia	zgon do 2 tyg. życia
pO_2/FiO_2	pO_2/FiO_2	pO_2/FiO_2	pO_2/FiO_2	pO_2/FiO_2
pO_2	pO_2	pO_2	pO_2	pO_2
FiO_2	FiO_2	FiO_2	FiO_2	FiO_2
HCO_3		HCO_3	HCO_3	HCO_3
pH	pH	pH	pH	pH
pCO_2	pCO_2	pCO_2	pCO_2	pCO_2
PDA				
IVH				
Odma				
Posocz				
NEC				
BE	BE	BE		
surf	surf	surf		
gage		gage		gage
płeć		płeć		płeć

Tablica 8.1: Zestawy danych

Tylko w pierwszym zestawie występują wszystkie parametry.

8.1.3 Parametr wyjściowy

Parametr wyjściowy to wystąpienie zgonu do końca 2 tygodnia życia.

8.1.4 Proces uczenia i testu

Proces obliczeniowy był przeprowadzony dla następujących dni hospitalizacji:
- 1 dzień hospitalizacji (24 godzina),
- 2 dzień hospitalizacji (48 godzina),
- 3 dzień hospitalizacji (72 godzina).

Badania najpierw przeprowadzono dla trzeciego dnia hospitalizacji, dla pięciu zestawów danych wejściowych.

Najlepszym, bo generującym najmniejszy błąd, okazał się zestaw 2, dla którego przeprowadzono badania w pierwszym i drugim dniu hospitalizacji.

Każdy zbiór pacjentów przed procesem uczenia i testu był 10-krotnie losowo podzielony na część uczącą i testową, dla których przeprowadzono obliczenia. Podział zbioru został dokonany w następujących proporcjach: 80% i 20%, odpowiednio dla zbioru uczącego i testowego. Uczenie metodą SVM przeprowadzono dla modelu C-SVC (C-Support Vector Classification) o funkcji jądra RBF: exp(-gamma*|u-v|^2) przy użyciu walidacji skośnej. Parametr C i parametr funkcji jądrowej (g - gamma) zostały dobrane automatycznie przy użyciu walidacji skośnej, osiągając niemal optymalne położenie, osiągając możliwie najwyższy poziom generalizacji.

8.2 Wyniki

W tablicach znajdują się wyniki klasyfikacji dla pierwszego, drugiego i trzeciego dnia hospitalizacji. Klasyfikacja polega na stwierdzeniu czy wystąpi zgon lub przeżycie pacjenta należącego do zbioru testowego.

8.2.1 Pierwszy dzień hospitalizacji (24 godzina)

Zbiór zawierał 43 pacjentów, w tym 12 zgonów do 14 doby życia.

Po losowym podziale zbioru (80% - część ucząca, 20% - część testowa), część ucząca obejmowała 34 pacjentów, a część testowa 9 pacjentów.

		PR			PR		PR [%] dla przeżycia		PR [%] dla zgonu		Wyniki	
Losowanie	Przeżycie	SVM	SSN	Zgon	SVM	SSN	SVM[%]	SSN[%]	SVM[%]	SSN[%]	SVM[%]	SSN[%]
1	7	7	7	2	1	1	100,0	100,0	50,0	50,0	88,9	88,9
2	5	5	5	4	3	3	100,0	100,0	75,0	75,0	88,9	88,9
3	7	7	7	2	1	1	100,0	100,0	50,0	50,0	88,9	88,9
4	6	4	6	3	2	2	66,7	100,0	66,7	66,7	66,7	88,9
5	8	6	7	1	1	1	75,0	87,5	100,0	100,0	77,8	88,9
6	7	5	7	2	2	2	71,4	100,0	100,0	100,0	77,8	100,0
7	6	5	5	3	2	2	83,3	83,3	66,7	66,7	77,8	77,8
8	6	6	6	3	3	2	100,0	100,0	100,0	66,7	100,0	88,9
9	7	7	7	2	0	1	100,0	100,0	0,0	50,0	77,8	88,9
10	7	6	7	2	1	1	85,7	100,0	50,0	50,0	77,8	88,9
Razem	**66**	**58**	**64**	**24**	**16**	**16**	**87,9**	**97,0**	**66,7**	**66,7**	**82,2**	**88,9**

Tablica 8.2: Wyniki klasyfikacji - 1 dzień hospitalizacji - Zestaw 2

Przyjęte oznaczenie dla zbiorów testowych i zbiorów uczących. Przeżycie oznacza stan, w którym pacjent przeżył określoną liczbę dni. W pracy przyjęto tę liczbę jako 14 dni. Zgon oznacza śmierć pacjenta w zadanym przedziale czasu. PR oznacza poprawne rozpoznanie zgonu, albo poprawne rozpoznanie przeżycia pacjenta za pomocą metod SVM oraz SSN.

8.2.2 Drugi dzień hospitalizacji (48 godzina)

Zbiór zawierał 22 pacjentów, w tym 8 zgonów do 14 doby życia.

Po losowym podziale zbioru (80% - część ucząca, 20% - część testowa), część ucząca obejmowała 17 pacjentów, a część testowa 5 pacjentów.

		PR			PR		PR [%] dla przeżycia		PR [%] dla zgonu		Wyniki	
Losowanie	Przeżycie	SVM	SSN	Zgon	SVM	SSN	SVM[%]	SSN[%]	SVM[%]	SSN[%]	SVM[%]	SSN[%]
1	3	2	3	2	1	1	66,7	100,0	50,0	50,0	60,0	80,0
2	5	4	5	0	-	-	80,0	100,0	-	-	80,0	100,0
3	1	1	1	4	2	3	100,0	100,0	50,0	75,0	60,0	80,0
4	3	3	2	2	1	1	100,0	66,7	50,0	50,0	80,0	60,0
5	4	4	4	1	0	0	100,0	100,0	0,0	0,0	80,0	80,0
6	3	3	3	2	1	0	100,0	100,0	50,0	0,0	80,0	60,0
7	3	3	3	2	1	1	100,0	100,0	50,0	50,0	80,0	80,0
8	2	1	2	3	2	1	50,0	100,0	66,7	33,3	60,0	60,0
9	4	3	4	1	1	1	75,0	100,0	100,0	100,0	80,0	100,0
10	3	3	3	2	1	1	100,0	100,0	50,0	50,0	80,0	80,0
Razem	**31**	**27**	**30**	**19**	**10**	**9**	**87,1**	**96,8**	**52,6**	**47,4**	**74,0**	**78,0**

Tablica 8.3: Wyniki klasyfikacji - 2 dzień hospitalizacji - Zestaw 2

8.2.3 Trzeci dzień hospitalizacji (72 godzina)

Zbiór zawierał 17 pacjentów, w tym 7 zgonów do 14 doby życia.

Po losowym podziale zbioru (80% - część ucząca, 20% - część testowa), część ucząca obejmowała 13 pacjentów, a część testowa 4 pacjentów.

W poniższych tablicach znajdują się wyniki testu dla 5 zestawów danych.

		PR			PR		PR [%] dla przeżycia		PR [%] dla zgonu		Wyniki	
Losowanie	**Przeżycie**	**SVM**	**SSN**	**Zgon**	**SVM**	**SSN**	**SVM[%]**	**SSN[%]**	**SVM[%]**	**SSN[%]**	**SVM[%]**	**SSN[%]**
1	3	3	3	1	0	1	100,0	100,0	0,0	100,0	75,0	100,0
2	1	1	1	3	1	1	100,0	100,0	33,3	33,3	50,0	50,0
3	3	2	3	1	1	1	66,7	100,0	100,0	100,0	75,0	100,0
4	3	3	2	1	1	1	100,0	66,7	100,0	100,0	100,0	75,0
5	3	1	2	1	1	1	33,3	66,7	100,0	100,0	50,0	75,0
6	2	2	1	2	1	2	100,0	50,0	50,0	100,0	75,0	75,0
7	3	3	3	1	0	1	100,0	100,0	0,0	100,0	75,0	100,0
8	2	2	2	2	1	1	100,0	100,0	50,0	50,0	75,0	75,0
9	3	2	2	1	1	1	66,7	66,7	100,0	100,0	75,0	75,0
10	3	2	3	1	1	1	66,7	100,0	100,0	100,0	75,0	100,0
Razem	**26**	**21**	**22**	**14**	**8**	**11**	**80,8**	**84,6**	**57,1**	**78,6**	**72,5**	**82,5**

Tablica 8.4: Wyniki klasyfikacji - 3 dzień hospitalizacji - Zestaw 1

		PR			PR		PR [%] dla przeżycia		PR [%] dla zgonu		Wyniki	
Losowanie	**Przeżycie**	**SVM**	**SSN**	**Zgon**	**SVM**	**SSN**	**SVM[%]**	**SSN[%]**	**SVM[%]**	**SSN[%]**	**SVM[%]**	**SSN[%]**
1	3	3	3	1	1	1	100,0	100,0	100,0	100,0	100,0	100,0
2	2	2	2	2	1	1	100,0	100,0	50,0	50,0	75,0	75,0
3	2	2	2	2	1	2	100,0	100,0	50,0	100,0	75,0	100,0
4	4	2	4	0	-	-	50,0	100,0	-	-	50,0	100,0
5	2	1	2	2	2	2	50,0	100,0	100,0	100,0	75,0	100,0
6	2	1	2	2	1	1	50,0	100,0	50,0	50,0	50,0	75,0
7	2	2	2	2	1	1	100,0	100,0	50,0	50,0	75,0	75,0
8	3	1	3	1	1	0	33,3	100,0	100,0	0,0	50,0	75,0
9	2	2	2	2	1	1	100,0	100,0	50,0	50,0	75,0	75,0
10	2	2	2	2	2	1	100,0	100,0	100,0	50,0	100,0	75,0
Razem	**24**	**18**	**24**	**16**	**11**	**10**	**75,0**	**100,0**	**68,8**	**62,5**	**72,5**	**85,0**

Tablica 8.5: Wyniki klasyfikacji - 3 dzień hospitalizacji - Zestaw 2

		PR			PR		PR [%] dla przeżycia		PR [%] dla zgonu		Wyniki	
Losowanie	**Przeżycie**	**SVM**	**SSN**	**Zgon**	**SVM**	**SSN**	**SVM[%]**	**SSN[%]**	**SVM[%]**	**SSN[%]**	**SVM[%]**	**SSN[%]**
1	2	2	2	2	0	1	100,0	100,0	0,0	50,0	50,0	75,0
2	1	1	1	3	0	2	100,0	100,0	0,0	66,7	25,0	75,0
3	3	2	3	1	0	1	66,7	100,0	0,0	100,0	50,0	100,0
4	2	2	2	2	0	1	100,0	100,0	0,0	50,0	50,0	75,0
5	2	2	2	2	1	1	100,0	100,0	50,0	50,0	75,0	75,0
6	1	1	1	3	1	1	100,0	100,0	33,3	33,3	50,0	50,0
7	3	1	3	1	1	0	33,3	100,0	100,0	0,0	50,0	75,0
8	3	2	3	1	1	1	66,7	100,0	100,0	100,0	75,0	100,0
9	4	1	4	0	-	-	25,0	100,0	-	-	25,0	100,0
10	3	3	3	1	0	1	100,0	100,0	0,0	100,0	75,0	100,0
Razem	**24**	**17**	**24**	**16**	**4**	**9**	**70,8**	**100,0**	**25,0**	**56,3**	**52,5**	**82,5**

Tablica 8.6: Wyniki klasyfikacji - 3 dzień hospitalizacji - Zestaw 3

		PR			PR		PR [%] dla przeżycia		PR [%] dla zgonu		Wyniki	
Losowanie	**Przeżycie**	**SVM**	**SSN**	**Zgon**	**SVM**	**SSN**	**SVM[%]**	**SSN[%]**	**SVM[%]**	**SSN[%]**	**SVM[%]**	**SSN[%]**
1	4	4	4	0	0	0	100,0	100,0	0,0	0,0	100,0	100,0
2	1	1	1	3	1	1	100,0	100,0	33,3	33,3	50,0	50,0
3	2	2	2	2	2	2	100,0	100,0	100,0	100,0	100,0	100,0
4	3	1	3	1	1	1	33,3	100,0	100,0	100,0	50,0	100,0
5	3	1	3	1	1	1	33,3	100,0	100,0	100,0	50,0	100,0
6	2	1	2	2	2	1	50,0	100,0	100,0	50,0	75,0	75,0
7	3	1	3	1	1	1	33,3	100,0	100,0	100,0	50,0	100,0
8	3	1	3	1	1	0	33,3	100,0	100,0	0,0	50,0	75,0
9	1	1	1	3	2	2	100,0	100,0	66,7	66,7	75,0	75,0
10	3	3	3	1	0	0	100,0	100,0	0,0	0,0	75,0	75,0
Razem	**25**	**16**	**25**	**15**	**11**	**9**	**64,0**	**100,0**	**73,3**	**60,0**	**67,5**	**85,0**

Tablica 8.7: Wyniki klasyfikacji - 3 dzień hospitalizacji - Zestaw 4

		PR			PR		PR [%] dla przeżycia		PR [%] dla zgonu		Wyniki	
Losowanie	Przeżycie	SVM	SSN	Zgon	SVM	SSN	SVM[%]	SSN[%]	SVM[%]	SSN[%]	SVM[%]	SSN[%]
1	2	1	2	2	1	0	50,0	100,0	50,0	0,0	50,0	50,0
2	3	3	3	1	1	1	100,0	100,0	100,0	100,0	100,0	100,0
3	0	-	-	4	0	2	-	-	0,0	50,0	0,0	50,0
4	3	1	3	1	1	1	33,3	100,0	100,0	100,0	50,0	100,0
5	3	3	3	1	1	1	100,0	100,0	100,0	100,0	100,0	100,0
6	3	2	3	1	1	1	66,7	100,0	100,0	100,0	75,0	100,0
7	3	3	3	1	0	1	100,0	100,0	0,0	100,0	75,0	100,0
8	2	1	2	2	1	1	50,0	100,0	50,0	50,0	50,0	75,0
9	2	0	2	2	1	0	0,0	100,0	50,0	0,0	25,0	50,0
10	3	2	3	1	0	0	66,7	100,0	0,0	0,0	50,0	75,0
Razem	**24**	**16**	**24**	**16**	**7**	**8**	**66,7**	**100,0**	**43,8**	**50,0**	**57,5**	**80,0**

Tablica 8.8: Wyniki klasyfikacji - 3 dzień hospitalizacji - Zestaw 5

8.3 Wnioski:

Metoda SVM pozwala klasyfikować pacjentów z poprawnością wyrażoną w procentach. Poniżej podano wyniki dla 1, 2 i 3 dnia hospitalizacji:

-1 dzień hospitalizacji: 82,2%

-2 dzień hospitalizacji: 74,0%

-3 dzień hospitalizacji: 72,5%.

Metoda SSN pozwala klasyfikować pacjentów z poprawnością wyrażoną w procentach. Poniżej podano wyniki dla 1, 2 i 3 dnia hospitalizacji:

-1 dzień hospitalizacji: 88,9%

-2 dzień hospitalizacji: 78,0%

-3 dzień hospitalizacji: 85,0%.

Największą poprawność klasyfikacji uzyskano dla 1 dnia hospitalizacji, gdy dostępna jest największa liczba pacjentów w zbiorze uczącym. Większą poprawność klasyfikacji uzyskano dzięki zastosowaniu metody SSN. Jednak generalnie metoda SVM ma przewagę polegającą na tym, że w sposób czytelny obrazuje objawy, w których znajduje się rozpatrywany przypadek. Pozwala to lekarzowi ocenić poprawne nastawy parametrów respiratora.

Rozdział 9

Podsumowanie

W pracy wykazano przydatność metod z dziedziny sztucznej inteligencji, lub bardziej precyzyjnie metod z zakresu inteligencji obliczeniowej, do rozwiązywania problemów związanych z zagadnieniami modelowania zjawiska niewydolności oddechowej u dzieci z zaburzeniami znanymi w literaturze jako RDS, Respiratory Distress Syndrome.

Metody użyte w pracy nie tylko dają efektywne narzędzie do badań stanów chorobowych ale także dostarczają narzędzia do badań metodami symulacji. Badania te umożliwiają określić przyszłe stany chorobowe. Metoda pozwala lekarzowi zbadać wpływ jego decyzji na stan chorego w najbliższej przyszłości, poprzez decyzje alternatywne.

W rozdziale 7 zawarto wyniki predykcji wskaźnika pO_2/FiO_2, natomiast w rozdziale 8 zamieszczono wyniki oceny ryzyka zgonu dla odpowiednich zestawów danych.

Predykcję wskaźnika pO_2/FiO_2 przeprowadzono w okresie trzech pierwszych dni hospitalizacji pacjenta.

Zdolności predykcyjne AIS zależą zarówno od parametrów sieci immunologicznej jak i od sposobu obliczania odległości (powinowactwa). Badania AIS przeprowadzono przy przyjęciu różnych definicji odległości. Zastosowano euklidesową miarę odległości oraz miarę odległości zdefiniowaną jako iloczyn skalarny. Mniejszy błąd predykcji uzyskano przy zastosowaniu odległości mierzonej jako iloczyn skalarny (AIS1).

Okres predykcji wynosił: 66, 60, 54, 48, 42, 36, 30, 24 [godz.] przy przyjęciu zakresu modelowania w oknie 72 godzin. Zdolność predykcyjna sztucznej sieci immunologicznej wzrasta wraz ze wzrostem podokna danych wejściowych (subwindow input), a tym samym zmniejszeniem podokna predykcji (subwindow output). Zdolność predykcyjna AIS wzrasta wraz ze zmniejszeniem okresu predykcji. Na przykład błąd predykcji dla 72 godziny wynosił 15,2 % dla podokna danych wejściowych (subwindow input) z obserwacji 6 godzinnych, a 8,7 % dla podokna danych wejściowych (subwindow input) z obserwacji 48 godzinnych.

Nawet wtedy gdy bezwzględne wartości błędu prognozy przekraczały maksymalny dopuszczalny błąd, trend zmian wskaźnika pO_2/FiO_2 prognozowany był właściwie.

Zdolność predykcji AIS porównano z predykcją modelu postępowania lekarza

MPL. Dla AIS i MPL zastosowano ten sam zbiór testowy.

W początkowym okresie podokna predykcji (subwindow input) błąd predykcji MPL jest mniejszy od AIS. Dla bardziej odległego horyzontu obserwacji błąd znacznie wzrasta i jest większy od wartości błędu predykcji metodą AIS.

W niektórych przypadkach model MPL nie znalazł w bazie danych dopasowania, czyli przypadków podobnych dla danego kryterium obliczania podobieństwa.

Metoda SVM pozwala klasyfikować pacjentów z poprawnością wyrażoną w procentach, odpowiednio dla pierwszego dnia hospitalizacji 82,2%, dla drugiego 74,0% oraz dla trzeciego 72,5%.

Metoda SSN pozwala klasyfikować pacjentów z poprawnością wyrażoną w procentach, odpowiednio dla pierwszego dnia hospitalizacji 88,9%, dla drugiego 78,0% oraz dla trzeciego 85,0%.

Największą poprawność klasyfikacji uzyskano dla 1 dnia hospitalizacji, gdy dostępna jest największa liczba pacjentów w zbiorze uczącym. Większą poprawność klasyfikacji uzyskano dzięki zastosowaniu metody SSN. Jednak generalnie metoda SVM ma przewagę polegającą na tym, że w sposób czytelny obrazuje objawy w których znajduje się rozpatrywany przypadek.

Spis tablic

Spis rysunków

Bibliografia

[1] David R. Musicant and O.L. Mangasarian. *Lagrangian Support Vector Machines.* http://ftp.cs.wisc.edu/math-prog/talks/nips00.ppt, December 1, 2000.

[2] De Casto, L. N., Von Zuben, F. J. (1999). *Artificial Immune Systems: Part 1 - Basic Theory and Applications.* Technical Report - RT DCA 01/99, p. 95.

[3] De Casto, L. N., Von Zuben, F. J. (2000a). *An Evolutionary Immune Network for Data Clustering.* Proc. of the IEEE SBRN, pp. 84-89.

[4] De Casto, L. N., Von Zuben, F. J. (2000b). *The Clonal Selection Algorithm with Engineering Applications.* GECCO'00 - Workshop Proceedings, pp. 36-37.

[5] Forrest, S., A. Perelson (1992). *Computation and the Immune System.* SIGBIO Newsletter, Association for Computing Machinery, 12(2), pp. 52-57.

[6] Forrest, S., A. Perelson, Allen, L., Cherukuri, R. (1994). *Self-Nonself Discrimination in a Computer.* Proc. of the IEEE Symposium on Research in Security and Privacy, pp. 202-212.

[7] Forrest, S., Hofmeyr S. A., Somayaji A. *Computer Immunology.* Communications of the ACM, 40(10), pp. 88-96. (1997).

[8] Jankowski N. *Ontogeniczne sieci neuronowe. O sieciach zmieniających swoją strukturę.* Warszawa : Akademicka Oficyna Wydawnicza RM, 1993.

[9] Kernel-Machines. *http://www.kernel-machines.org.*

[10] Kruczek P. *Ocena przydatności prognostycznej sztucznej sieci neuronowej u urodzonych przedwcześnie noworodków z zespołem zaburzeń oddychania.* Rozprawa doktorska 2001.

[11] Kruczek P., Pietrzyk J.J., Sukiennik A., Wajs W. *Blood gases values forecasting by artificial neural network in prematurely born infants with respiratory distress.* Przegląd Lekarski, 2002, 59 Suppl 1 34-7.

[12] Kwinta P., Kruczek P., Stoch P., Wajs W., Pietrzak J. *Result of continuous monitoring of hemoglobin saturation during the first month of life as predictors of retinopathy of prematurity.* Pediatric Research 2005, 58 390.

[13] LIBSVM - A Library for Support Vector Machines, C-C. Chang and C-J. Lin [Online]. *http://www.csie.ntu.edu.tw/ cjlin/libsvm.*

[14] LSVM - Lagrangian Support Vector Machine . *http://www.cs.wisc.edu/dmi/lsvm/.*

[15] Mangasarian O. L. *http://jmlr.csail.mit.edu/papers/volume1/mangasarian01a/ mangasarian01a.pdf.*

[16] Rutkowski L. *Metody i techniki sztucznej inteligencji.* WN PWN, Wyd. 1, 2006.

[17] Specht D.F. *Probabilistic Neural Networks.* Neural Networks, 1990.

[18] Staines N., Brostoff J. James K. *Wprowadzenie do immunology.* Wydanie I, Urban & Partner, Wrocław 1998.

[19] Sukiennik A. *Sieci neuronowe jako narzędzie wykorzystane do prognozy wartości parametrów gazometrii krwi.* rozprawa doktorska 2002.

[20] SVM - Support Vector Machines. *http://www.support-vector-machines.org/.*

[21] Tadeusiewicz R. *Elementarne wprowadzenie do techniki sieci neuronowych z przykładowymi programami.* Oficyna Wydawnicza, Warszawa, 1998.

[22] Tadeusiewicz R. *Sieci neuronowe.* Warszawa : Akademicka Oficyna Wydawnicza RM, 1993.

[23] Vapnik V. *The Nature of Statistical Learning Theory.* Springer-Verlag 1995 New York.

[24] Vapnik V. *The support vector method of function estimation.* C.M. Bishop, redaktor, Neural Networks and Machines Learning, s. 239-268. Springer-Verlag, 1998.

[25] Wais P. ,Wajs W., Święcicki M. *Zastosowanie Sieci Immunologicznej do Klasyfikacji Szeregów Czasowych.* Sztuczna Inteligencja w Inżynierii Biomedycznej, SIIB 2004.

[26] Wajs W., Wais P., Maj G., Święcicki M., Sukiennik A., Kruczek P., Pietrzyk J.J., Stoch P. *Classification of vectorized medical data sets using artificial immune algorithms.* Proceedings of IFAC workshop on Programmable Devices and Systems PDS 2004,395-400.

[27] Wajs W., Wais P., Święcicki M.,Wojtowicz H. *Artificial Immune System for medical data classification.* International Conference on Computational Science, ICCS 2005, p. 810, Springer-Verlag, 2005.

[28] Wajs W. Święcicki M., Wais P. *An Artificial immune algorithms apply to pre-processing signals.* International Conference on Computational Science, ICCS 2004, pp. 703-710, Springer-Verlag, 2004.

[29] Wajs W., Święcicki M.,Wais P.,Wojtowicz H., Janik P. *Zastosowanie sieci immunologicznej do drążenia danych.* Ogólnopolska Konferencja Naukowa, Modelowanie Cybernetyczne Systemów Biologicznych, MCSB 2005.

[30] Wajs W., Święcicki M.,Wais P.,Wojtowicz H., Janik P.,Nowak L. *Predictive Analysis of Blood Gasometry Parameters Related to the Infants Respiration Insufficiency.* International Conference on Computational Science ,ICCS 2006, pp.300-307, Springer Berlin, Heidelberg, 2006.

[31] Wajs W., Święcicki M.,Wais P.,Wojtowicz H., Janik P.,Nowak L. *Predictive Analysis of the pO2 Blood Gasometry Parameter Related to the Infants Respiration Insufficiency.* Intelligent Information Systems 2006, IIS 2006, pp. 377-381, Springer-Verlag, 2006.

[32] Wierzchoń S. T. *Sztuczne systemy immunologiczne - teoria i zastosowanie.* Akademicka Oficyna Wydawnicza EXIT, Warszawa 2001.

[33] Wolfram MathWorld. *http://mathworld.wolfram.com/LagrangeMultiplier.html.*

Dodatek A

Dodatek A - Kody użytych funkcji

A.1 Sztuczne Sieci Immunologiczne - AIS - algorytm nr 1

Segmentacja zbioru przed nauką AIS

```
%zbior -  zbiór do nauki

%w - liczba wierszy (pacjentów)
%m - liczba pomiarów na 1 parametr (zbiór wejściowy)
%p - liczba parametrów gazometrii krwi
%sz_ok - szerokosc okna czasowego

[w,k]=size(zbior);

S=[];
for i=1:w,
   for j=1:m-sz_ok+1                        % przesuwanie okna czas.
      S1=zbior(i,j:j+sz_ok-1);              % dla 1 parametru
      S2=zbior(i,j+m:j+m+sz_ok-1);          % dla 2 parametru
      S3=zbior(i,j+(2*m):j+(2*m)+sz_ok-1);  % dla 3 parametru
      S4=zbior(i,j+(3*m):j+(3*m)+sz_ok-1);  % dla 4 parametru

      A=[S1,S2,S3,S4];
      S=[S;A];
   end
end

% Ag zbior do nauki AIS
Ags=S;
save Ags Ags;
```

Nauka AIS - Funkcja: ainet_sun.m

```
function [M,D] = ainet_sun(VAg,ts,n,N,gen,qi,tp,pa,ma)
%
% W oryginalnym kodzie zmieniono:
% - funkcje obliczania odległośći pomiędzy antygenami
%   (nowe funkcje: odl, odleglosc)
% - funkcję suppress
% - clone
% - re-selection
%
% Ph.D. Thesis
% Leandro Nunes de Castro
% February, 2000
%
% Internal functions: CLONE, SUPPRESS, VER_EQ, EXTRACT, NORMA
%
% M     -> memory cells matrix
% D     -> distance matrix for M
% Ag    -> antigens (training patterns)
% Ab    -> network antibodies
% n     -> no. of best-matching cells taken for each Ag (Selection)
```

```
% N     -> clone number multiplier
% gen   -> maximum number of generations
%
% L     -> Ag and Ab length
% N1    -> no. of antibodies (constructive)
% N2    -> no. of antigens
% Nc    -> no. of clones to be generated
% D     -> Ag-Ab affinity vector
% Do    -> D sorted in ascending order
% mi    -> learning (hypermutation) rate (default: 4.0)
% qi    -> percentile amount of clones to be Re-selected
% ts    -> suppression threshold (default: 0.001)
% tp    -> pruning threshold
% D1    -> idiotypic affinity matrix [N1,N1]
% vbD   -> vector best affinity for each Ag
% nR    -> no. of Ab to be re-selected
%
% Adequate parameters for solving the SPIRAL task: td = 0.1; ts = 0.04; gen = 100;

% pa - ilość parametrów wejściowych
% ma - szerokosc okna czasowego (ilosc pomiarow dla jednego parametru wej.)

[Ab, Ag ]=MakeAb( VAg, 0.5);

[N2,L] = size(Ag);
N1 = 10;
mi = 4.0;
sc = 0.001;

disp(sprintf('Suppression threshold: %5.4f',ts));
disp(sprintf('Pruning threshold: %5.4f',tp/100));
disp(sprintf('Number of best matching cells to be selected: %d',n));
disp(sprintf('Percentile amount of clones to be re-selected: %d%%',100*qi));
disp(sprintf('Number of generations: %d',gen));
disp(sprintf('Population (to be recognized) size: [%d,%d]',N2,L));
disp('Press any key to continue...');

pause(2);

it = 0; avD = 1000;
RUNNET = 0;
while it < gen & avD > sc,
!echo "it" it > prog.log
   vbD = []; M = []; i = 1;
   vet = randperm(N2);

   while i <= N2,

     % Ag-Ab Affinity (Match-Function)
     % Nowa odległość - obliczana względem każdego parametru wejściowego
      D=[0];k=0;
      Dpom=[0];
      while k<pa,
        Dk = odl(Ab(:,(k*ma+1):(k+1)*ma),Ag(vet(i),(k*ma+1):(k+1)*ma));
         D=D+Dk;
         Dpom=D;
        k=k+1;
      end;

      [Dn,I] = sort(norma(D));
      Nc = floor(N-Dn(1:n,:)*N);

      % Clone & Affinity Maturation
      % Nowe klonowanie - względem każdego parametru wejściowego
      Cpom=[0];C=[]; k=0;
      while k<pa,
        [Ck,Cag,Cmi] = clone(Ab(:,(k*ma+1):(k+1)*ma),Ag(vet(i),(k*ma+1):(k+1)*ma),mi,D,I,Nc);
        Ck = Ck - Cmi.*(Ck-Cag);
        C=cat(2,C,Ck);
        Cpom=C;
        k=k+1;
      end;

        % Re-Selection
        % Nowa re-selekcja - względem każdego parametru wejściowego
        D=[0];k=0;
        while k<pa,
          Dk = odl(C(:,(k*ma+1):(k+1)*ma),Ag(vet(i),(k*ma+1):(k+1)*ma));
         D=D+Dk;
```

```
        Dpom=D;
       k=k+1;
     end;

     [Dn,I] = sort(D);
     nR = round(qi*size(C,1));
     m = C(I(1:nR),:); % 1 clone for each Ag
     D = D(I(1:nR));   % new affinities

     % Network pruning (Natural Death)
     Ip = find(D > tp);
     m = extract(m,Ip); D = extract(D,Ip);

     % Suppression (Idiotypic Network)
     [m,D1] = suppress(m,ts,pa,ma);

     % General parameters
     minD = min(D); [vbD] = [vbD; minD];
     Cs = size(m,1);
     M = [M; m];  % memory matrix
     i = i + 1;
  end;

   % Search for similaritites among clusters
   [M,D] = suppress(M,ts,pa,ma);

   % Re-build Ab repertoire
   Ab = [M;rand(N1,L)];

   if( ~isempty( vbD ) )
      avD = [avD;mean(vbD)];
   else
      disp('.')
   end

   it = it + 1;

   disp(sprintf('It: %d  tp %4.3f avD: %f   Net size: %d',it, tp ,avD(end),size(M,1)));
end; %while it < gen & avD > sc

%END FUNCTION AINET


% Function clone %

function [C,Cag,Cmi] = clone(Ab,ag,mi,D,I,Nc);
% C   -> clones (from greater to smaller affinity)
% Cag -> clones of Ag
% Cmi -> clones of mi
% S   -> selected antibodies
% Obs.: Cag and Cmi are necessary for the updating procedure
%       The original cell is mantained
[N1,L] = size(Ab); [n,N2] = size(Nc);

C = Ab(I(1),:); Cmi = ones(1,L); Cag = C; % Maintenance of the fittest cell before maturation
for i=1:n,
   vones = ones(Nc(i),1);
   C = [C; vones * Ab(I(i),:)];
   Cag = [Cag; vones * ag];
   Cmi = [Cmi; rand(Nc(i),L) .* D(I(i)) .* mi];
end;

% Function suppress self-recognizing and non-stimulated Ab from Memory (M)
function [M,D1] = suppress(M,ts,pa,ma);
% M   -> memory matrix
% D1  -> idiotypic affinity matrix
D1=[0];k=0;
while k<pa,
   % Nowa odległość - obliczana względem każdego parametru wejściowego
   Dk = odleglosc(M(:,(k*ma+1):(k+1)*ma),M(:,(k*ma+1):(k+1)*ma));
   D1=D1+Dk;
   k=k+1;
end;

aux = triu(D1,1);
[Is,Js] = find(aux>0 & aux<ts);
if ~isempty(Is),
   Is = ver_eq(Is);
   M = extract(M,Is);
```

```
   % D1 = extract(D1,Is);
end;


D1=[0];k=0;
while k<pa,
   % Nowa odległość - obliczana względem każdego parametru wejściowego
   Dk = odleglosc(M(:,(k*ma+1):(k+1)*ma),M(:,(k*ma+1):(k+1)*ma));
   D1=D1+Dk;
   k=k+1;
end;


% Search for repeated indexes
function [Is] = ver_eq(I);
l = length(I); Is = [];
if l > 1,
   for i=1:l-1,
      aux = I(i);
      auxI = I(i+1:end);
      el = find(auxI == aux);
      if isempty(el),
         Is = [Is,aux];
      end;
   end;
   Is = [Is,I(end)];
else,
   Is = I;
end;

% Function Extracts lines from M indexed by I
function [M] = extract(M,I);
Maux = zeros(size(M));
Maux(I,:) = M(I,:);
M = M - Maux;
[I] = find(M(:,1)~=0);
M = M(I,:);

% Function normalizes matrix over [0,1]
function [Dn] = norma(D);
% Dn  -> normalized vector over [0,1]
[np,ni] = size(D);
if ni == 1,
   Dn = (D - min(D))./(max(D)-min(D));
else,
   vmaxD = max(D); vminD = min(D);
   for i=1:ni,
      Dn(:,i) = (D(:,i) - vminD(i))./(vmaxD(i)-vminD(i));
   end;
end;
% End Function NORMA
%####################################################################

function [Ab, Agt]=MakeAb( Ag, p )

[m,n]=size(Ag);
I=randperm(m);
m2=round(m*(p));

%tworzenie zbioru trenującego (m2% zbioru Ag)
Ab=zeros(m2,n);
for i=1:m2
    Ab(i,:)=Ag(I(i),:);
end;

%tworzenie zbioru p-cial
Agt=zeros(m-m2,n);
for i=m2+1:m
    Agt(i-m2,:)=Ag(I(i) , : ) ;
end;

%end function MakeAb
%####################################################################
```

Odległość pomiędzy antygenami - Funkcja: odleglosc.m

Obliczenia prowadzone w środowisku rozproszonym MDC.

```
function z = odleglosc(w,p)
% w,p  - macierze antygenów

[w_M, w_N]=size(w);
[p_M, p_N]=size(p);

%!echo "Funkcja odleglosc"
if( p_M >= 300)

 % !echo "Funkcja odleglosc - p_M >300"
  jm = findResource('jobmanager', 'Name', 'asyipol1')
  job1=createJob(jm);
  set(job1,'FileDependencies', {'/home/piotrek/funkcje'})

  z = odlska(w,p,job1);
  destroy(job1);

elseif( p_M < 300)
   z = odl(w,p);
end
%end function
%#######################################

function z = odlska(w,p,job1)
%
[w_M, w_N]=size(w);
[p_M, p_N]=size(p);

%w_M
%p_M
w=Normalizuj(w);
p=Normalizuj(p);

if( p_M > 1)

   z=[];
   t={};
   krok=300;
   j=1;
   k=1;
   licz_krok = floor(p_M/krok);
   cala = licz_krok * krok;

   for i=1:krok:cala
      q=p(i:i+krok-1,:);
      q=q';
      t{j} = createTask(job1, @ilo, 1, {w q});
      j=j+1;
   end
   for i=i+krok:i+krok
      q=p(i:p_M,:);
      q=q';
      t{j} = createTask(job1, @ilo, 1, {w q});
      j=j+1;
   end

% !echo "Przygotowane zadania"

 submit(job1)

% !echo "Skończone submit"
 waitForState(job1)

% !echo "Koniec funkcji wait"

      for k=1:j-1
       results = get(t{k}, 'OutputArguments');
       size(results{1})
       z=[z, results{1}];
       size(z)
      end

% !echo "Macierz z załadowana"

      for k=1:j-1
       destroy(t{k})
      end

     % destroy(job1)
```

```
%!echo "Zadania - wyczyszczone"

elseif( p_M == 1)
   z=w*p';
elseif( isempty(p) & isempty(w))
   z=[];
end

eps=0.001;
z=1./(z+eps);
z=abs(1-z);
%###################################

function [Y]=Normalizuj(X)
[X_M, X_N]=size(X);
Y=X;
for i=1:X_M
  Y(i, :)=X(i, :)/norm( X(i, :));
end
%end function
%###################################

function z = odl(w,p)
[w_M, w_N]=size(w);
[p_M, p_N]=size(p);

w=Normalizuj(w);
p=Normalizuj(p);

if( p_M > 1)
   z=[];
   for i=1:p_M
      q=p(i,:);
      z=[z, w*q'];
   end

elseif( p_M == 1)
   z=w*p';
elseif( isempty(p) & isempty(w))
   z=[];
end
eps=0.001;
z=1./(z+eps);
z=abs(1-z);
%end function odl
%###################################

function z = ilo(a,b)
z=a*b;
%end function
```

Odpowiedź AIS - Funkcja: Chk_answer

```
function [Odp]= Chk_answer(Ag, M1, ts1, pa, ma, mp)

% pa -liczba parametrów wejściowych
% ma - szerokość okna czasowego (liczba pomiarów na jeden parametr)
% mp - szerokość podokna danych wej.
% mr - różnica szerokości okien mr=ma-mp;
% mpp = mp - pomocnicze do Showresult

[M_Ag, N_Ag]=size(Ag);

mpp=mp;

for i=1:M_Ag,
   mr=ma-mp;
   mrx=mr;
   if( any( Ag(i, : ) ) )

      Agp1(1,1:mp)=Ag(i,1:mp);
      Agp2(1,1:mp)=Ag(i,mp+1:2*mp);
      Agp3(1,1:mp)=Ag(i,2*mp+1:3*mp);
      Agp4(1,1:mp)=Ag(i,3*mp+1:4*mp);
      %Agp5(1,1:mp)=Ag(i,4*mp+1:5*mp);
      %Agp6(1,1:mp)=Ag(i,5*mp+1:6*mp);
```

```
      Agpt=[Agp1,Agp2,Agp3,Agp4];
      %Agpt=[Agp1];

      for j=1:mrx

        [V1, vbD1, Dn1 ] = AnswerAIN( M1, ts1, ( Agpt(1, : )), pa,ma,mr );

      mp=mp+1;
      mr=ma-mp;

      Agp1(1,mp)=V1(1,mp);
      Agp2(1,mp)=V1(1,ma+mp);
      Agp3(1,mp)=V1(1,2*ma+mp);
      Agp4(1,mp)=V1(1,3*ma+mp);
     % Agp5(1,mp)=V1(1,3*ma+mp);
     % Agp6(1,mp)=V1(1,3*ma+mp);

     Agpt=[Agp1,Agp2,Agp3,Agp4];
     %Agpt=[Agp1];

    end;

   [V1, vbD1, Dn1 ] = AnswerAIN( M1, ts1, ( Agpt(1, : )), pa,ma,mr );

   ShowResult( (Ag(i, :)), V1(:, :),ma ,mpp)

   end

   pause
end

Odp=V1(1,:);
%end fuction chk_answer
%#################################################################

function [M, vbD, Dn ] = AnswerAIN( M, ds, Ag, pa, ma, mr )

vbD=[];
Dk=[];
% pa -liczba parametrów gazometrii krwi
% ma - szerokość okna czasowego (liczba pomiarów na jeden parametr)
% mp - szerokość podokna danych wej.
% mr - różnica szerokości okien mr=ma-mp;

D=[0];k=0;
% Nowa odległość - obliczana względem każdego parametru wejściowego
while k<pa
   Dk = odl(M(:,(k*ma+1):((k+1)*ma)-mr),Ag(:,((k*ma+1))-k*mr:((k+1)*ma)-((k+1)*mr) ));
   D=D+Dk;
   k=k+1;
end;

[Dn,I] = sort(norma(D));

J=find(Dn < ds );

Dn=Dn( J );

vbD=M(I(J), : );

M=M(I, :);

%end function AnswerAIN
%#########################################################################

% Function Extracts lines from M indexed by I
function [M] = extract(M,I);
Maux = zeros(size(M));
Maux(I,:) = M(I,:);
M = M - Maux;
[I] = find(M(:,1)~=0);
M = M(I,:);

%end function extract
%#########################################################################

% Function normalizes matrix over [0,1]
function [Dn] = norma(D);
% Dn  -> normalized vector over [0,1]
```

```
[np,ni] = size(D);
if ni == 1,
   Dn = (D - min(D))./(max(D)-min(D));
else,
   vmaxD = max(D); vminD = min(D);
   for i=1:ni,
      Dn(:,i) = (D(:,i) - vminD(i))./(vmaxD(i)-vminD(i));
   end;
end;
% End Function NORMA
%##########################################################################

 function ShowResult( Wzor, Answer, ma, mpp)
clf

 plot( Wzor(:,1:mpp), 'k-')
 hold on

 plot( Answer(1,1:ma) ,'k--' )
 hold on
 grid

% end function
```

A.2 Sztuczne Sieci Immunologiczne - AIS - algorytm nr 2

Nauka AIS - Funkcja: siec.m

```
%dane - zbiór do nauki; load dane
%Agt_m - zbiór do testu; load Agt_m
%wybranie do testu dwóch parametrów:
%pO2/FiO2(wsk) i pH
wsk=dane(1:120,1:4);
ph=dane(1:120,30:33);
baza=[wsk(1:120,:),ph(1:120,:)];

% tworzę Ab i Ag dla parametru wsk
Abzw=wsk(1:120,:);
Agw=Agt_m(8,1:3);

% tworzę Ab i Ag dla parametru ph
Abzp=ph(1:120,:);
Agp=Agt_m(8,5:7);

% MMwsk - macierz mutantów dla wsk
MMwsk=imm(Agw,Abzw);

% MMph - macierz mutantów dla ph
MMph=imm(Agp,Abzp);

% MM - macierz mutantów (wektory w parach: wsk,ph)
MM=[MMwsk,MMph];

% Obliczanie odległości dla i-tej pary wektorów do całej bazy
% pa - ilość parametrów w wektorze
% ma - ilość pomiarów na parametr
% w - ilość par wektorów
% D - wektor odległości
[w,z]=size(MM);
Dp=0; wyn=[];
for i=1:w
 pa=2;
 ma=4;
 D=[0];k=0;
 while k<pa,
  Dk = dist(baza(:,(k*ma+1):(k+1)*ma),MM(i,(k*ma+1):(k+1)*ma)');
  D=D+Dk;
  k=k+1;
 end;
 [x y]=min(D)
 Dp(i)=x;
 wyn=[wyn;baza(y,:)]
end;
```

```
%wybór z bazy wektora o najmniejszej odległości
[x y]=min(Dp)
wynik=wyn(y,:);
%##########################################################################
function [MM] = imm(Ag,Abz)
% Abz-zbiór przeciwciał
% Ag-antygen
% n-powtórzeń
n=10;

%losowy wybór przeciwciał
%I-indeksy losowo wybranych przeciwciał
[k,l]=size(Abz);
I=randperm(k)
pr=0; C=[];
while pr<n,
 Ip=I(4*pr+1:4*pr+4)

 %tworzenie zbioru Ab z 4 przypadków
 for i=1:4
  Ab(i,:)=Abz(Ip(i),:);
 end;

 %obliczanie odległości: funkcja odl
 D=[];
 [k,l]=size(Ab);
 for i=1:k,
  D(i,:) = odl(Ag,Ab(i,:));
 end;
 [x y]=min(D)

 % macierz C* złożona z n wektorów
 C=[C;Ab(y,:)]
 pr=pr+1;
end;

%obliczamy min i max
mn=min(C(:,4))
mx=max(C(:,4))

% mutacja poprzez losowanie, losujemy m razy
% przedział mutacji: (mn + (mx-mn) * rand(1))
m=5;

% MM - macierz mutantów Ag
MM=[];
for i=1:m
 Agp=[Ag,(mn + (mx-mn) * rand(1))];
 MM=[MM;Agp];
end;
%##########################################################################
%funkcja odległość
function [d] = odl(ag,ab)
[m,n]=size(ag);
[k,l]=size(ab);
i=1; d=0; dp=0;
while i<=n,
 j=1;
 while j<=l,
  %dp=ag(i)*ab(j);
  dp=abs(ag(i)-ab(j));
  %dp=sqrt(ag(i)*ab(j));
  %dp=(ag(i)-ab(j)).*(ag(i)-ab(j));
  d=d+dp;
  j=j+1;
 end;
 d=sqrt(d);
 d=1/(3*4) * d;
 i=i+1;
end;
%end function odl
```

A.3 Support Vector Machines -SVM

Skrypt easy.py

```
#!/usr/bin/env python

import sys
import os

if len(sys.argv) <= 1:
        print 'Usage: %s training_file [testing_file]' % sys.argv[0]
        raise SystemExit

# svm, grid, and gnuplot executable

is_win32 = (sys.platform == 'win32')
if not is_win32:
        svmscale_exe = "../svm-scale"
        svmtrain_exe = "../svm-train"
        svmpredict_exe = "../svm-predict"
        grid_py = "./grid.py"
        gnuplot_exe = "/usr/bin/gnuplot"
else:
        # example for windows
        svmscale_exe = r"..\svm\svmscale.exe"
        svmtrain_exe = r"..\svm\svmtrain.exe"
        svmpredict_exe = r"..\svm\svmpredict.exe"
        gnuplot_exe = r"c:\tmp\gnuplot\bin\pgnuplot.exe"
        grid_py = r"..\svm\grid.py"

assert os.path.exists(svmscale_exe),"svm-scale executable not found"
assert os.path.exists(svmtrain_exe),"svm-train executable not found"
assert os.path.exists(svmpredict_exe),"svm-predict executable not found"
assert os.path.exists(gnuplot_exe),"gnuplot executable not found"
assert os.path.exists(grid_py),"grid.py not found"

train_pathname = sys.argv[1]
assert os.path.exists(train_pathname),"training file not found"
file_name = os.path.split(train_pathname)[1]
scaled_file = file_name + ".scale"
model_file = file_name + ".model"
range_file = file_name + ".range"

if len(sys.argv) > 2:
        test_pathname = sys.argv[2]
        file_name = os.path.split(test_pathname)[1]
        assert os.path.exists(test_pathname),"testing file not found"
        scaled_test_file = file_name + ".scale"
        predict_test_file = file_name + ".predict"

cmd = "%s -s %s %s > %s" % (svmscale_exe, range_file, train_pathname, scaled_file)
print 'Scaling training data...'
os.system(cmd)

cmd = "%s -svmtrain %s -gnuplot %s %s" % (grid_py, svmtrain_exe, gnuplot_exe, scaled_file)
print 'Cross validation...'
dummy, f = os.popen2(cmd)

line = ''
while 1:
        last_line = line
        line = f.readline()
        if not line: break
c,g,rate = map(float,last_line.split())

print 'Best c=%s, g=%s CV rate=%s' % (c,g,rate)

cmd = "%s -c %s -g %s %s %s" % (svmtrain_exe,c,g,scaled_file,model_file)
print 'Training...'
os.popen(cmd)

print 'Output model: %s' % model_file
if len(sys.argv) > 2:
        cmd = "%s -r %s %s > %s" % (svmscale_exe, range_file, test_pathname, scaled_test_file)
        print 'Scaling testing data...'
        os.system(cmd)

        cmd = "%s %s %s %s" % (svmpredict_exe, scaled_test_file, model_file, predict_test_file)
```

```
        print 'Testing...'
        os.system(cmd)

        print 'Output prediction: %s' % predict_test_file
```

Plik: svm-train.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "svm.h"
#define Malloc(type,n) (type *)malloc((n)*sizeof(type))

void exit_with_help()
{
        printf(
        "Usage: svm-train [options] training_set_file [model_file]\n"
        "options:\n"
        "-s svm_type : set type of SVM (default 0)\n"
        "       0 -- C-SVC\n"
        "       1 -- nu-SVC\n"
        "       2 -- one-class SVM\n"
        "       3 -- epsilon-SVR\n"
        "       4 -- nu-SVR\n"
        "-t kernel_type : set type of kernel function (default 2)\n"
        "       0 -- linear: u'*v\n"
        "       1 -- polynomial: (gamma*u'*v + coef0)^degree\n"
        "       2 -- radial basis function: exp(-gamma*|u-v|^2)\n"
        "       3 -- sigmoid: tanh(gamma*u'*v + coef0)\n"
        "       4 -- precomputed kernel (kernel values in training_set_file)\n"
        "-d degree : set degree in kernel function (default 3)\n"
        "-g gamma : set gamma in kernel function (default 1/k)\n"
        "-r coef0 : set coef0 in kernel function (default 0)\n"
        "-c cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)\n"
        "-n nu : set the parameter nu of nu-SVC, one-class SVM, and nu-SVR (default 0.5)\n"
        "-p epsilon : set the epsilon in loss function of epsilon-SVR (default 0.1)\n"
        "-m cachesize : set cache memory size in MB (default 100)\n"
        "-e epsilon : set tolerance of termination criterion (default 0.001)\n"
        "-h shrinking: whether to use the shrinking heuristics, 0 or 1 (default 1)\n"
        "-b probability_estimates: whether to train a SVC or SVR model for probability estimates,
          0 or 1 (default 0)\n"
        "-wi weight: set the parameter C of class i to weight*C, for C-SVC (default 1)\n"
        "-v n: n-fold cross validation mode\n"
        );
        exit(1);
}

void parse_command_line(int argc, char **argv, char *input_file_name, char *model_file_name);
void read_problem(const char *filename);
void do_cross_validation();

struct svm_parameter param;             // set by parse_command_line
struct svm_problem prob;                // set by read_problem
struct svm_model *model;
struct svm_node *x_space;
int cross_validation;
int nr_fold;

int main(int argc, char **argv)
{
        char input_file_name[1024];
        char model_file_name[1024];
        const char *error_msg;

        parse_command_line(argc, argv, input_file_name, model_file_name);
        read_problem(input_file_name);
        error_msg = svm_check_parameter(&prob,&param);

        if(error_msg)
        {
                fprintf(stderr,"Error: %s\n",error_msg);
                exit(1);
        }

        if(cross_validation)
        {
```

```
                do_cross_validation();
        }
        else
        {
                model = svm_train(&prob,&param);
                svm_save_model(model_file_name,model);
                svm_destroy_model(model);
        }
        svm_destroy_param(&param);
        free(prob.y);
        free(prob.x);
        free(x_space);

        return 0;
}

void do_cross_validation()
{
        int i;
        int total_correct = 0;
        double total_error = 0;
        double sumv = 0, sumy = 0, sumvv = 0, sumyy = 0, sumvy = 0;
        double *target = Malloc(double,prob.l);

        svm_cross_validation(&prob,&param,nr_fold,target);
        if(param.svm_type == EPSILON_SVR ||
           param.svm_type == NU_SVR)
        {
                for(i=0;i<prob.l;i++)
                {
                        double y = prob.y[i];
                        double v = target[i];
                        total_error += (v-y)*(v-y);
                        sumv += v;
                        sumy += y;
                        sumvv += v*v;
                        sumyy += y*y;
                        sumvy += v*y;
                }
                printf("Cross Validation Mean squared error = %g\n",total_error/prob.l);
                printf("Cross Validation Squared correlation coefficient = %g\n",
                        ((prob.l*sumvy-sumv*sumy)*(prob.l*sumvy-sumv*sumy))/
                        ((prob.l*sumvv-sumv*sumv)*(prob.l*sumyy-sumy*sumy))
                        );
        }
        else
        {
                for(i=0;i<prob.l;i++)
                        if(target[i] == prob.y[i])
                                ++total_correct;
                printf("Cross Validation Accuracy = %g%%\n",100.0*total_correct/prob.l);
        }
        free(target);
}

void parse_command_line(int argc, char **argv, char *input_file_name,
                        char *model_file_name)
{
        int i;

        // default values
        param.svm_type = C_SVC;
        param.kernel_type = RBF;
        param.degree = 3;
        param.gamma = 0;        // 1/k
        param.coef0 = 0;
        param.nu = 0.5;
        param.cache_size = 100;
        param.C = 1;
        param.eps = 1e-3;
        param.p = 0.1;
        param.shrinking = 1;
        param.probability = 0;
        param.nr_weight = 0;
        param.weight_label = NULL;
        param.weight = NULL;
        cross_validation = 0;

        // parse options
```

```
for(i=1;i<argc;i++)
{
        if(argv[i][0] != '-') break;
        if(++i>=argc)
                exit_with_help();
        switch(argv[i-1][1])
        {
                case 's':
                        param.svm_type = atoi(argv[i]);
                        break;
                case 't':
                        param.kernel_type = atoi(argv[i]);
                        break;
                case 'd':
                        param.degree = atoi(argv[i]);
                        break;
                case 'g':
                        param.gamma = atof(argv[i]);
                        break;
                case 'r':
                        param.coef0 = atof(argv[i]);
                        break;
                case 'n':
                        param.nu = atof(argv[i]);
                        break;
                case 'm':
                        param.cache_size = atof(argv[i]);
                        break;
                case 'c':
                        param.C = atof(argv[i]);
                        break;
                case 'e':
                        param.eps = atof(argv[i]);
                        break;
                case 'p':
                        param.p = atof(argv[i]);
                        break;
                case 'h':
                        param.shrinking = atoi(argv[i]);
                        break;
                case 'b':
                        param.probability = atoi(argv[i]);
                        break;
                case 'v':
                        cross_validation = 1;
                        nr_fold = atoi(argv[i]);
                        if(nr_fold < 2)
                        {
                                fprintf(stderr,"n-fold cross validation: n must >= 2\n");
                                exit_with_help();
                        }
                        break;
                case 'w':
                        ++param.nr_weight;
                        param.weight_label = (int *)realloc(param.weight_label,
                                                        sizeof(int)*param.nr_weight);
                        param.weight = (double *)realloc(param.weight,
                                                        sizeof(double)*param.nr_weight);
                        param.weight_label[param.nr_weight-1] = atoi(&argv[i-1][2]);
                        param.weight[param.nr_weight-1] = atof(argv[i]);
                        break;
                default:
                        fprintf(stderr,"unknown option\n");
                        exit_with_help();
        }
}

// determine filenames

if(i>=argc)
        exit_with_help();

strcpy(input_file_name, argv[i]);

if(i<argc-1)
        strcpy(model_file_name,argv[i+1]);
else
{
        char *p = strrchr(argv[i],'/');
```

```
                if(p==NULL)
                        p = argv[i];
                else
                        ++p;
                sprintf(model_file_name,"%s.model",p);
        }
}

// read in a problem (in svmlight format)

void read_problem(const char *filename)
{
        int elements, max_index, i, j;
        FILE *fp = fopen(filename,"r");

        if(fp == NULL)
        {
                fprintf(stderr,"can't open input file %s\n",filename);
                exit(1);
        }

        prob.l = 0;
        elements = 0;
        while(1)
        {
                int c = fgetc(fp);
                switch(c)
                {
                        case '\n':
                                ++prob.l;
                                // fall through,
                                // count the '-1' element
                        case ':':
                                ++elements;
                                break;
                        case EOF:
                                goto out;
                        default:
                                ;
                }
        }
out:
        rewind(fp);

        prob.y = Malloc(double,prob.l);
        prob.x = Malloc(struct svm_node *,prob.l);
        x_space = Malloc(struct svm_node,elements);

        max_index = 0;
        j=0;
        for(i=0;i<prob.l;i++)
        {
                double label;
                prob.x[i] = &x_space[j];
                fscanf(fp,"%lf",&label);
                prob.y[i] = label;

                while(1)
                {
                        int c;
                        do {
                                c = getc(fp);
                                if(c=='\n') goto out2;
                        } while(isspace(c));
                        ungetc(c,fp);
                        if (fscanf(fp,"%d:%lf",&(x_space[j].index),&(x_space[j].value)) < 2)
                        {
                                fprintf(stderr,"Wrong input format at line %d\n", i+1);
                                exit(1);
                        }
                        ++j;
                }
out2:
                if(j>=1 && x_space[j-1].index > max_index)
                        max_index = x_space[j-1].index;
                x_space[j++].index = -1;
        }

        if(param.gamma == 0)
```

```
                param.gamma = 1.0/max_index;

        if(param.kernel_type == PRECOMPUTED)
                for(i=0;i<prob.l;i++)
                {
                        if (prob.x[i][0].index != 0)
                        {
                                fprintf(stderr,"Wrong input format: first column must be 0
                                                                        :sample_serial_number\n");
                                exit(1);
                        }
                        if ((int)prob.x[i][0].value <= 0 || (int)prob.x[i][0].value > max_index)
                        {
                                fprintf(stderr,"Wrong input format:
                                                sample_serial_number out of range\n");
                                exit(1);
                        }
                }

        fclose(fp);
}
```

Plik: svm-predict.c

```
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>
#include "svm.h"

char* line;
int max_line_len = 1024;
struct svm_node *x;
int max_nr_attr = 64;

struct svm_model* model;
int predict_probability=0;

void predict(FILE *input, FILE *output)
{
        int correct = 0;
        int total = 0;
        double error = 0;
        double sumv = 0, sumy = 0, sumvv = 0, sumyy = 0, sumvy = 0;

        int svm_type=svm_get_svm_type(model);
        int nr_class=svm_get_nr_class(model);
        int *labels=(int *) malloc(nr_class*sizeof(int));
        double *prob_estimates=NULL;
        int j;

        if(predict_probability)
        {
                if (svm_type==NU_SVR || svm_type==EPSILON_SVR)
                        printf("Prob. model for test data: target value = predicted value
                               + z,\nz: Laplace distribution e^(-|z|/sigma)/(2sigma),
                               sigma=%g\n",svm_get_svr_probability(model));
                else
                {
                        svm_get_labels(model,labels);
                        prob_estimates = (double *) malloc(nr_class*sizeof(double));
                        fprintf(output,"labels");
                        for(j=0;j<nr_class;j++)
                                fprintf(output," %d",labels[j]);
                        fprintf(output,"\n");
                }
        }
        while(1)
        {
                int i = 0;
                int c;
                double target,v;

                if (fscanf(input,"%lf",&target)==EOF)
                        break;

                while(1)
```

```
                {
                        if(i>=max_nr_attr-1)    // need one more for index = -1
                        {
                                max_nr_attr *= 2;
                                x = (struct svm_node *)
                                    realloc(x,max_nr_attr*sizeof(struct svm_node));
                        }

                        do {
                                c = getc(input);
                                if(c=='\n' || c==EOF) goto out2;
                        } while(isspace(c));
                        ungetc(c,input);
                        if (fscanf(input,"%d:%lf",&x[i].index,&x[i].value) < 2)
                        {
                                fprintf(stderr,"Wrong input format at line %d\n", total+1);
                                exit(1);
                        }
                        ++i;
                }

out2:
                x[i++].index = -1;

                if (predict_probability && (svm_type==C_SVC || svm_type==NU_SVC))
                {
                        v = svm_predict_probability(model,x,prob_estimates);
                        fprintf(output,"%g ",v);
                        for(j=0;j<nr_class;j++)
                                fprintf(output,"%g ",prob_estimates[j]);
                        fprintf(output,"\n");
                }
                else
                {
                        v = svm_predict(model,x);
                        fprintf(output,"%g\n",v);
                }

                if(v == target)
                        ++correct;
                error += (v-target)*(v-target);
                sumv += v;
                sumy += target;
                sumvv += v*v;
                sumyy += target*target;
                sumvy += v*target;
                ++total;
        }
        if (svm_type==NU_SVR || svm_type==EPSILON_SVR)
        {
                printf("Mean squared error = %g (regression)\n",error/total);
                printf("Squared correlation coefficient = %g (regression)\n",
                       ((total*sumvy-sumv*sumy)*(total*sumvy-sumv*sumy))/
                       ((total*sumvv-sumv*sumv)*(total*sumyy-sumy*sumy))
                       );
        }
        else
                printf("Accuracy = %g%% (%d/%d) (classification)\n",
                       (double)correct/total*100,correct,total);
        if(predict_probability)
        {
                free(prob_estimates);
                free(labels);
        }
}

void exit_with_help()
{
        printf(
        "Usage: svm-predict [options] test_file model_file output_file\n"
        "options:\n"
        "-b probability_estimates: whether to predict probability estimates,
          0 or 1 (default 0); for one-class SVM only 0 is supported\n"
        );
        exit(1);
}

int main(int argc, char **argv)
{
```

```
	FILE *input, *output;
	int i;

	// parse options
	for(i=1;i<argc;i++)
	{
		if(argv[i][0] != '-') break;
		++i;
		switch(argv[i-1][1])
		{
			case 'b':
				predict_probability = atoi(argv[i]);
				break;
			default:
				fprintf(stderr,"unknown option\n");
				exit_with_help();
		}
	}
	if(i>=argc)
		exit_with_help();

	input = fopen(argv[i],"r");
	if(input == NULL)
	{
		fprintf(stderr,"can't open input file %s\n",argv[i]);
		exit(1);
	}

	output = fopen(argv[i+2],"w");
	if(output == NULL)
	{
		fprintf(stderr,"can't open output file %s\n",argv[i+2]);
		exit(1);
	}

	if((model=svm_load_model(argv[i+1]))==0)
	{
		fprintf(stderr,"can't open model file %s\n",argv[i+1]);
		exit(1);
	}

	line = (char *) malloc(max_line_len*sizeof(char));
	x = (struct svm_node *) malloc(max_nr_attr*sizeof(struct svm_node));
	if(predict_probability)
		if(svm_check_probability_model(model)==0)
		{
			fprintf(stderr,"Model does not support probabiliy estimates\n");
			exit(1);
		}
	predict(input,output);
	svm_destroy_model(model);
	free(line);
	free(x);
	fclose(input);
	fclose(output);
	return 0;
}
```

Plik: svm-scale.c

```
/*
	scale attributes to [lower,upper]
	usage: scale [-l lower] [-u upper] [-y y_lower y_upper]
		     [-s filename] [-r filename] filename
*/
#include <float.h>
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>

char *line;
int max_line_len = 1024;
double lower=-1.0,upper=1.0,y_lower,y_upper;
int y_scaling = 0;
double *feature_max;
double *feature_min;
```

```
double y_max = -DBL_MAX;
double y_min = DBL_MAX;
int max_index;

#define max(x,y) ((x>y)?x:y)
#define min(x,y) ((x<y)?x:y)

void output_target(double value);
void output(int index, double value);
char* readline(FILE *input);

int main(int argc,char **argv)
{
        int i,index;
        FILE *fp;
        char *save_filename = NULL;
        char *restore_filename = NULL;

        for(i=1;i<argc;i++)
        {
                if(argv[i][0] != '-') break;
                ++i;
                switch(argv[i-1][1])
                {
                        case 'l': lower = atof(argv[i]); break;
                        case 'u': upper = atof(argv[i]); break;
                        case 'y':
                                y_lower = atof(argv[i]);
                                ++i;
                                y_upper = atof(argv[i]);
                                y_scaling = 1;
                                break;
                        case 's': save_filename = argv[i]; break;
                        case 'r': restore_filename = argv[i]; break;
                        default:
                                fprintf(stderr,"unknown option\n");
                                exit(1);
                }
        }

        if(!(upper > lower) || (y_scaling && !(y_upper > y_lower)))
        {
                fprintf(stderr,"inconsistent lower/upper specification\n");
                exit(1);
        }

        if(argc != i+1)
        {
                fprintf(stderr,"usage: %s [-l lower] [-u upper] [-y y_lower y_upper]\n",argv[0]);
                fprintf(stderr,"      [-s save_filename] [-r restore_filename] filename\n");
                fprintf(stderr,"(default: lower = -1, upper = 1, no y scaling)\n");
                exit(1);
        }

        fp=fopen(argv[i],"r");

        if(fp==NULL)
        {
                fprintf(stderr,"can't open file %s\n", argv[i]);
                exit(1);
        }

        line = (char *) malloc(max_line_len*sizeof(char));

#define SKIP_TARGET\
        while(isspace(*p)) ++p;\
        while(!isspace(*p)) ++p;

#define SKIP_ELEMENT\
        while(*p!=':') ++p;\
        ++p;\
        while(isspace(*p)) ++p;\
        while(*p && !isspace(*p)) ++p;

        /* assumption: min index of attributes is 1 */
        /* pass 1: find out max index of attributes */
        max_index = 0;
        while(readline(fp)!=NULL)
        {
```

```
        char *p=line;

        SKIP_TARGET

        while(sscanf(p,"%d:%*f",&index)==1)
        {
                max_index = max(max_index, index);
                SKIP_ELEMENT
        }
}

feature_max = (double *)malloc((max_index+1)* sizeof(double));
feature_min = (double *)malloc((max_index+1)* sizeof(double));

if(feature_max == NULL || feature_min == NULL)
{
        fprintf(stderr,"can't allocate enough memory\n");
        exit(1);
}

for(i=0;i<=max_index;i++)
{
        feature_max[i]=-DBL_MAX;
        feature_min[i]=DBL_MAX;
}

rewind(fp);

/* pass 2: find out min/max value */
while(readline(fp)!=NULL)
{
        char *p=line;
        int next_index=1;
        double target;
        double value;

        sscanf(p,"%lf",&target);
        y_max = max(y_max,target);
        y_min = min(y_min,target);

        SKIP_TARGET

        while(sscanf(p,"%d:%lf",&index,&value)==2)
        {
                for(i=next_index;i<index;i++)
                {
                        feature_max[i]=max(feature_max[i],0);
                        feature_min[i]=min(feature_min[i],0);
                }

                feature_max[index]=max(feature_max[index],value);
                feature_min[index]=min(feature_min[index],value);

                SKIP_ELEMENT
                next_index=index+1;
        }

        for(i=next_index;i<=max_index;i++)
        {
                feature_max[i]=max(feature_max[i],0);
                feature_min[i]=min(feature_min[i],0);
        }
}

rewind(fp);

/* pass 2.5: save/restore feature_min/feature_max */

if(restore_filename)
{
        FILE *fp = fopen(restore_filename,"r");
        int idx, c;
        double fmin, fmax;

        if(fp==NULL)
        {
                fprintf(stderr,"can't open file %s\n", restore_filename);
                exit(1);
        }
```

```
        if((c = fgetc(fp)) == 'y')
        {
                fscanf(fp, "%lf %lf\n", &y_lower, &y_upper);
                fscanf(fp, "%lf %lf\n", &y_min, &y_max);
                y_scaling = 1;
        }
        else
                ungetc(c, fp);

        if (fgetc(fp) == 'x') {
                fscanf(fp, "%lf %lf\n", &lower, &upper);
                while(fscanf(fp,"%d %lf %lf\n",&idx,&fmin,&fmax)==3)
                {
                        if(idx<=max_index)
                        {
                                feature_min[idx] = fmin;
                                feature_max[idx] = fmax;
                        }
                }
        }
        fclose(fp);
}

if(save_filename)
{
        FILE *fp = fopen(save_filename,"w");
        if(fp==NULL)
        {
                fprintf(stderr,"can't open file %s\n", save_filename);
                exit(1);
        }
        if(y_scaling)
        {
                fprintf(fp, "y\n");
                fprintf(fp, "%.16g %.16g\n", y_lower, y_upper);
                fprintf(fp, "%.16g %.16g\n", y_min, y_max);
        }
        fprintf(fp, "x\n");
        fprintf(fp, "%.16g %.16g\n", lower, upper);
        for(i=1;i<=max_index;i++)
        {
                if(feature_min[i]!=feature_max[i])
                        fprintf(fp,"%d %.16g %.16g\n",i,feature_min[i],feature_max[i]);
        }
        fclose(fp);
}

/* pass 3: scale */
while(readline(fp)!=NULL)
{
        char *p=line;
        int next_index=1;
        int index;
        double target;
        double value;

        sscanf(p,"%lf",&target);
        output_target(target);

        SKIP_TARGET

        while(sscanf(p,"%d:%lf",&index,&value)==2)
        {
                for(i=next_index;i<index;i++)
                        output(i,0);

                output(index,value);

                SKIP_ELEMENT
                next_index=index+1;
        }

        for(i=next_index;i<=max_index;i++)
                output(i,0);

        printf("\n");
}

free(line);
```

```
        fclose(fp);
        return 0;
}

char* readline(FILE *input)
{
        int len;

        if(fgets(line,max_line_len,input) == NULL)
                return NULL;

        while(strrchr(line,'\n') == NULL)
        {
                max_line_len *= 2;
                line = (char *) realloc(line, max_line_len);
                len = strlen(line);
                if(fgets(line+len,max_line_len-len,input) == NULL)
                        break;
        }
        return line;
}

void output_target(double value)
{
        if(y_scaling)
        {
                if(value == y_min)
                        value = y_lower;
                else if(value == y_max)
                        value = y_upper;
                else value = y_lower + (y_upper-y_lower) *
                             (value - y_min)/(y_max-y_min);
        }
        printf("%g ",value);
}

void output(int index, double value)
{
        /* skip single-valued attribute */
        if(feature_max[index] == feature_min[index])
                return;

        if(value == feature_min[index])
                value = lower;
        else if(value == feature_max[index])
                value = upper;
        else
                value = lower + (upper-lower) *
                        (value-feature_min[index])/
                        (feature_max[index]-feature_min[index]);

        if(value != 0)
                printf("%d:%g ",index, value);
}
```

Plik: svm.h

```
#ifndef _LIBSVM_H
#define _LIBSVM_H

#ifdef __cplusplus
extern "C" {
#endif

struct svm_node
{
        int index;
        double value;
};

struct svm_problem
{
        int l;
        double *y;
        struct svm_node **x;
};
```

```
enum { C_SVC, NU_SVC, ONE_CLASS, EPSILON_SVR, NU_SVR }; /* svm_type */
enum { LINEAR, POLY, RBF, SIGMOID, PRECOMPUTED }; /* kernel_type */

struct svm_parameter
{
        int svm_type;
        int kernel_type;
        int degree;     /* for poly */
        double gamma;   /* for poly/rbf/sigmoid */
        double coef0;   /* for poly/sigmoid */

        /* these are for training only */
        double cache_size; /* in MB */
        double eps;     /* stopping criteria */
        double C;       /* for C_SVC, EPSILON_SVR and NU_SVR */
        int nr_weight;          /* for C_SVC */
        int *weight_label;      /* for C_SVC */
        double* weight;         /* for C_SVC */
        double nu;      /* for NU_SVC, ONE_CLASS, and NU_SVR */
        double p;       /* for EPSILON_SVR */
        int shrinking;  /* use the shrinking heuristics */
        int probability; /* do probability estimates */
};

struct svm_model *svm_train(const struct svm_problem *prob,
                            const struct svm_parameter *param);
void svm_cross_validation(const struct svm_problem *prob,
                          const struct svm_parameter *param,
                          int nr_fold, double *target);

int svm_save_model(const char *model_file_name,
                   const struct svm_model *model);
struct svm_model *svm_load_model(const char *model_file_name);

int svm_get_svm_type(const struct svm_model *model);
int svm_get_nr_class(const struct svm_model *model);
void svm_get_labels(const struct svm_model *model, int *label);
double svm_get_svr_probability(const struct svm_model *model);

void svm_predict_values(const struct svm_model *model,
                        const struct svm_node *x, double* dec_values);
double svm_predict(const struct svm_model *model, const struct svm_node *x);
double svm_predict_probability(const struct svm_model *model,
                               const struct svm_node *x, double* prob_estimates);

void svm_destroy_model(struct svm_model *model);
void svm_destroy_param(struct svm_parameter *param);

const char *svm_check_parameter(const struct svm_problem *prob,
                                const struct svm_parameter *param);
int svm_check_probability_model(const struct svm_model *model);

#ifdef __cplusplus
}
#endif

#endif /* _LIBSVM_H */
```

A.4 Sztuczne Sieci Neuronowe - SSN

```
%Klasyfikacja ryzyka zgonu
%Oznaczenia klasy:
%1 - przeżycie
%2 - zgon

% P  - Wejscie do nauki
% Tc - Wyjscie do nauki
% Pt - Wejscie do testu
% Yc - Wyjscie z testu
% Realc - prawdziwe wartoci klas z testu:

P=P';
Pt=Pt';
T = ind2vec(Tc)

sp = 0.3
net = newpnn(P,T,sp);
Y = sim(net,P)
Yc = vec2ind(Y)
Y = sim(net,Pt)
Yc = vec2ind(Y)
```

A.5 Przygotowanie danych z bazy danych dla modelu SVM

```
import dbs.tools.*;
import java.io.*;
class gxx
{
public static void main(String args[]) { new Filtr(); }
public gxx(){ w1(); }
public void w1()
{ int u=0;
try
{
//Otworzenie pliku, do którego będą zapisywane wektory
//do dalszej analizy FileOutputStream
fs = new FileOutputStream("f:\\plik_do_analizy.txt");
PrintStream ds = new PrintStream(fs);

//Połączenie z bazą danych
Sql out=new Sql();
out.openODBC("db1");

// Wczytanie numerów id wszystkich pacjentów w bazie danych
String patid[]=getColumnValues(out, "gazometria_krwi", "patient");

// Sprawdź każdego pacjenta z bazy danych
for(int ii=0; ii<patid.length; ii++)
{ int czas[]=new int[1000];
float po2_fio2[]=new float[1000];
float fio2[] = new float[1000];
float po2[]=new float[1000];
float ph[]=new float[1000];
float pco2[]=new float[1000];
float hco3[]=new float[1000];
float be[]=new float[1000];
// Wczytanie wartości parametrów gazometrii
out.executeQuery("select patient, po2, fio2,ph,pco2,hco3,czas,
be from gazometria_krwi, gazometria_krwi_TMP_Time
where gazometria_krwi.id=gazometria_krwi_TMP_Time.id
and patient="+patid[ii]  +" order by czas");
int max_row=out.getRowCount();
// Jeżli liczba pomiarów gazometrii jest <=4 to przerwij analizę
// i przejdź do następnego pacjenta
if(max_row>4)
{
// Wczytanie wartości parametrów gazometrii
out.beforeFirst();
for(int i=0; i<out.getRowCount(); i++)
{
out.next();
czas[i]=out.getInt(7);
po2_fio2[i]=out.getFloat(2)/out.getFloat(3);
fio2[i]=out.getFloat(3);
```

```
po2[i]=out.getFloat(2);
ph[i]=out.getFloat(4);
pco2[i]=out.getFloat(5);
hco3[i]=out.getFloat(6);
be[i]=out.getFloat(8);
}
// Wczytanie masy urodzeniowej pacjenta
int birthweight=0;
out.executeQuery("select birthweight from perinatal where patient="+patid[ii]);
out.first();
birthweight=out.getInt(1);

// Wczytanie godziny podania surfactantu
int med[]=null;
out.executeQuery("select medications, czas from med_dtl, med_dtl_TMP_Time where
med_dtl.id=med_dtl_TMP_Time.id and patient="+patid[ii] +" and (medications=237
or medications=152 or medications=151  or medications=345) order by czas");
med=readMed(out, czas);

// Sprawdzenie czy pacjent był podłączony do respiratora imv lub simv
// i wykonano dla niego przynajmniej 5 odczytów tych paramatrów
int f=0;
out.executeQuery("select patient, czas from simv,simv_TMP_Time
where simv.id=simv_TMP_Time.id and
czas <= 72 and patient="+patid[ii]  +" order by czas");

if(out.getRowCount()>5)f++;
out.executeQuery("select patient, czas from imv,imv_TMP_Time
where imv.id=imv_TMP_Time.id and
czas <= 72 and patient="+patid[ii]  +" order by czas");

if(out.getRowCount()>5)f++;
if(f>0)
{
// Wyliczenie godziny zgonu pacjenta
// licząc od chwili przyjęcia na oddział long zgon=0;
out.executeQuery("select distinct admissions.adm_date,
admissions.adm_time, dis.hospital_dod, dis.time_of_discharge,
admissions.patient from admissions, dis  where
admissions.patient=dis.patient
and admissions.admission=dis.admission
and dis.hospital_status='Z'
and admissions.patient="+patid[ii]);
zgon=readZgon(out);
int godzina_analizy=24;

// Jeżeli analizowany wektor obejmuje
//czas życia pacjenta co najmniej 24 godziny
if(check(czas)>=godzina_analizy)
{
//zamiana godziny zgonu pacjenta na wartości -1, 1,
//gdzie -1 oznacza zgon w pierwszych 2 tygodniach życia
if(zgon<=336&&zgon>0)zgon=-1;
else zgon=1;

//Warunek końcowy po2/fio2 w analizowanej godzinie <=100
//oraz masa urodzeniowa <= 1500
if(po2_fio2[godzina_analizy]<=100 && birthweight<=1500)
{ds.print(zgon+"\t");
ds.print(po2_fio2[godzina_analizy]+"\t");
ds.print(po2[godzina_analizy]+"\t");
ds.print(fio2[godzina_analizy]+"\t");
ds.print(hco3[godzina_analizy]+"\t");
ds.print(ph[godzina_analizy]+"\t");
ds.print(pco2[godzina_analizy]+"\t");
ds.print(getMed(med, godzina_analizy)+"\t");

//getMed(int[], int) zwraca ile razy został podany surfaktant
//w czasie nie przekraczającym godziny w której dane będą analizowane
}
}
}
}
}
ds.close();
}
catch(Exception e){}
}
}
```