Grupa projektowa: **WT, TP, 7:30** Wrocław, 5 lutego 2017r.

Piotr Chmiel, 200608
Kamil Machnicki, 200752
Łukasz Matysiak, 200646
Grzegorz Zając, 200664
Jakub Zgraja, 200609

Miękkie metody obliczeniowe

DOKUMENTACJA PROJEKTOWA

1. SYSTEM ROZPOZNAWANIA OPARTY O SIECI BAYESOWSKIE

2. SYSTEM ROZPOZNAWANIA OPARTY O UKRYTY MODEL MARKOWA

Rok akad. 2016/2017, kierunek INF, specjalność IMT, studia II stopnia

PROWADZĄCY:
dr inż. Konrad Jackowski

Spis treści

1 Wstęp teoretyczny

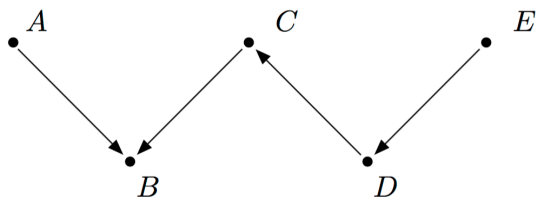
1.1 Sieci Bayesowskie

Definicja Sieci Bayesowskiej

Siecią Bayesa nazywamy skierowany graf acykliczny o wierzchołkach reprezentujących zmienne losowe i łukach określających zależności. Istnienie łuku pomiędzy dwoma wierzchołkami oznacza istnienie bezpośredniej zależności przyczynowo skutkowej pomiędzy odpowiadającymi im zmiennymi. Siła tej zależności określona jest przez tablice prawdopodobieństw warunkowych.

Inaczej mówiąc, sieć Bayesowska to acykliczny (nie zawierający cykli) graf skierowany, w którym:

- Węzły reprezentują zmienne losowe (np. temperaturę jakiegoś źródła, stan pacjenta, cechę obiektu itp.)
- Łuki (skierowane) reprezentują zależność typu „zmienna X ma bezpośredni wpływ na zmienna Y",
- Każdy węzeł X ma stowarzyszona z nim tablice prawdopodobieństw warunkowych określających wpływ wywierany na X przez jego poprzedników (rodziców) w grafie.
- Zmienne reprezentowane przez węzły przyjmują wartości dyskretne.



Rysunek 1: Przykładowa sieć Bayesowska

Na rysunku został przedstawiony przykład nieskomplikowanej sieci Bayesowskiej. Na podstawie zaprezentowanego na rysunku grafu przedstawiającego sieć można wyciągnąć następujące wnioski:

- Para zmiennych A oraz B jest od siebie bezpośrednio zależna.
- Para zmiennych B oraz C jest od siebie bezpośrednio zależna.
- Zmienna reprezentowana przez wierzchołek B jest jednocześnie zależna od A oraz C.
- Zmienne A oraz C pozostają brzegowo niezależne do momentu ustalenia wartości B (własność tą można również określić mówiąc, że zmienne A i C są d-połączone).
- Zmienne E oraz C są zależne pośrednio.
- Zmienna D d-separuje zmienne C i E.
- Zmienne C i D d-separują parę zmiennych B, E.

Orientacja łuków jest konieczna do określenia zależności nieprzechodnich. Na przytoczonym przykładzie można zaobserwować, że pomimo faktu przemienności własności zależności orientacja krawędzi wnosi istotną informację o rozkładzie.

Sieć Bayesowska pozwala na wyznaczenie rozkładu prawdopodobieństwa zmiennych. Jeżeli przez Π(Xi) oznaczymy zbiór rodziców danego wierzchołka w grafie, to rozkład prawdopodobieństwa zmiennych danej sieci opisuje się równaniem:

$$P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i|\Pi(X_i))$$

Co dla przedstawionego powyżej przykładu wynosi:

$$P(A, B, C, D, E) = P(A)P(B|A, C)P(C|D)P(D|E)P(E)$$

Dzięki powyższemu równaniu równania możliwe jest określenie prawdopodobieństwa wystąpienia określonego wartościowania wszystkich zmiennych, znając jedynie lokalne prawdopodobieństwa warunkowe. Określając wartości tzw. *przyczyn podstawowych*, czyli węzłów grafu nie posiadających rodziców (zmienne A, E w podanym przykładzie), można określić wartości oczekiwane innych atrybutów. Wszystkie pozostałe zmienne zależą bowiem pośrednio lub bezpośrednio od tego zbioru.

Istotą działania sieci Bayesowskich jest propagacja informacji o rozkładzie prawdopodobieństwa. W praktyce jednak **sieci Bayesowskie używane są do ekstrakcji informacji o rozkładach nieznanych, o których wiadomości możemy czerpać tylko z dostępnych wyników prób z eksperymentów statystycznych.** Sieci te nie są zatem używane do opisywania dokładnych rozkładów.

Próby z eksperymentów statystycznych (zestawy danych uczących) mogą mieć bardzo duże rozmiary, a ekstrakcja zawartych w nich w nich informacji może być zadaniem bardzo złożonym obliczeniowo. Proces uczenia sieci może być zatem postrzegany jako statystyczna kompresja informacji o rozkładzie do zwartej i jednocześnie bardziej przydatnej do wnioskowania bayesowskiego postaci.

Konstruowanie sieci Bayesowskiej składa się z następujących kroków:

- zdefiniowanie zmiennych,
- zdefiniowanie połączeń pomiędzy zmiennymi,
- określenie prawdopodobieństw warunkowych *a priori*
- wprowadzenie danych do sieci,
- uaktualnienie sieci,
- wyznaczenie prawdopodobieństw *a posteriori*

Uczenie sieci jest uczeniem bez nadzoru i bez wstępnej wiedzy eksperckiej, a co za tym idzie zakłada się, że na wstępie (bez znajomości próby statystycznej) wszystkie dozwolone struktury sieci są jednakowo prawdopodobne, a przy ustalonej strukturze grafu wszystkie możliwe poprawne zbiory tablic prawdopodobieństw warunkowych są jednakowo prawdopodobne.

1.1.1 Algorytm Chow-Liu

Algorytm Chow-Liu stanowi klasyczny algorytm odtwarzający kształt zależności w próbie. Algorytm nie buduje sieci Bayesowskiej, a jedynie niezorientowane drzewo zależności. Jeżeli sieć Bayesa danego rozkładu ma postać drzewa, to algorytm Chow-Liu poprawnie odtworzy jego kształt. Algorytm Chow-Liu nie sprawdza, czy rozkład zmiennych zadanej próby D spełnia powyższy warunek. Jeśli struktura zależności między zmiennymi w próbie nie jest drzewem, to algorytm znajduje najlepszą strukturę drzewiastą, która opisuje postać zależności. Należy jednak pamiętać, że w skrajnie niekorzystnych wypadkach informacja zwrócona przez algorytm może być błędna.

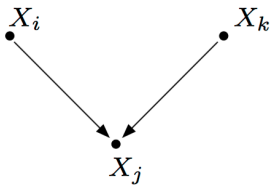
Przebieg algorytmu:

Krok 1: Niech G oznacza graf pełny o zbiorze wierzchołków tożsamym ze zbiorem atrybutów próby D. Wówczas niech T będzie nieskierowaną strukturą drzewiastą uzyskaną w wyniku zastosowania dowolnego algorytmu znajdowania minimalnego drzewa rozpinającego (MST) w grafie G z funkcją wagową określającą stopień zależności między zmiennymi.

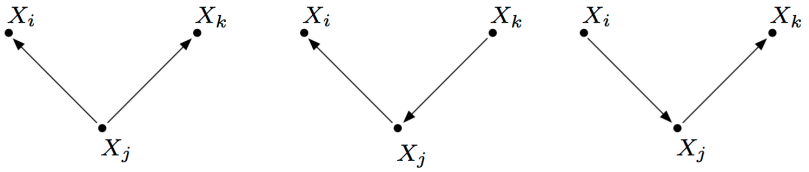
Krok 2: W drzewie T należy obrać kierunki w sposób dowolny, uzyskując wynikową strukturę sieci Bayesa BS.

W pierwszym kroku możliwe jest wykorzystanie odległości Kullback-Leiblera jako funkcji wagowej:

$$DEP(X_i, X_j) = \sum_{x_i, x_j} P(x_i, x_j) log \frac{P(x_i, x_j)}{P(x_i)P(x_j)}$$



Rysunek 2: Dwu rodziców: $DEP(X_i, X_k) = 0 \wedge DEP(X_i, X_k|X_j) > 0$



Rysunek 3: X_j d-separuje $X_i, X_k : DEP(X_i, X_k) > 0 \wedge DEP(X_i, X_k|X_j) = 0$

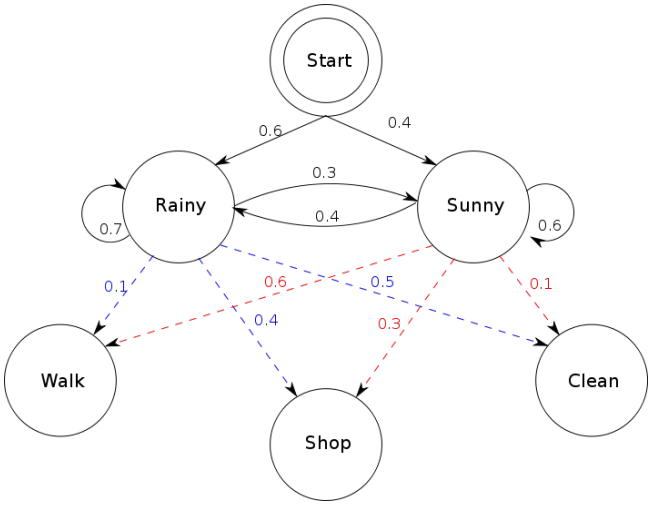
Gdzie podobnie jak w warunku małe litery x_i, x_j oznaczają wartościowania zmiennych X_i, X_ja sumowanie przebiega po wszystkich wartościowaniach w próbie D.

1.2 Ukryte modele Markowa

1.2.1 Wstęp

Ukryte modele Markowa (ang. *Hidden Markov Model*, w skrócie *HMM*) jest określeniem zaawansowanego modelu statystycznego znajdującego obecnie zastosowanie w wielu dziedzinach inżynierskich, szczególnie tam, gdzie analizowane są zjawiska o charakterze sekwencji losowych zdarzeń jak na przykład mowa czy gesty. Termin ten został wprowadzony i opisany matematycznie w drugiej połowie lat sześćdziesiątych ubiegłego wieku przez Bauma i Petriego, zaś swą nazwę zawdzięcza podstawie matematycznej, na której się opiera - łańcuchowi Markowa, który to określony został przez rosyjskiego matematyka Andrieja Markowa.

W rozumieniu ogólnym ukryte modele Markowa określają system zdolny z pewnym prawdopodobieństwem przewidzieć jak zachowa się modelowany obiekt w przyszłości, bazując tylko i wyłącznie na jego aktualnym stanie, gdyż nie jest przechowywana historia wyników, jakie osiągał w przeszłości. Cały system dzieli się na niewidoczne dla obserwatora ukryte stany oraz część obserwowaną (wyjście), która jest losową funkcją stanu.



Rysunek 4: Ogólny wygląd modelu Markowa.

Dzięki swej rozbudowanej matematycznej charakterystyce, ukryte modele Markowa sukcesywnie stosowane są jako baza teoretyczna do rozwiązywania wielu problemów, a poprawnie zastosowane w praktyce dają bardzo dobre rezultaty. Głównymi zagadnieniami, przy których są one stosowane są modele akustyczne przy rozpoznawaniu mowy, rozpoznawanie pisma ręcznego, obiektów czy gestów. Znajdują zastosowanie również szeroko w bioinformatyce, biomedycynie, w psychologii do modelowania procesów uczenia się, w finansach przy modelowaniu ryzyka na rynku obligacji, przy prognozowaniu pogody, do generowania muzyki czy do wypełniania brakujących wyrazów w zdaniach. Jako ciekawe zastosowania wymienić również można modelowanie erupcji gejzeru *Old Faithful* czy zbudowanie bota *Mark V. Shaney* podszywającego się pod zwykłego użytkownika *Usenetu* w latach osiemdziesiątych.

Jednym z głównych problemów występujących przy budowanie takiego modelu jest określenie odpowiedniego układu poprzez wyznaczenie topologii i dobranie wartości parametrów tak, aby otrzymać jak największą efektywność przy rozpoznawaniu. Klasyczne metody doboru parametrów modelu, jak algorytm *Bauma-Welcha*, czy metody gradientowe, nie zapewniają znalezienia optymalnych wartości oraz wymagają poczynienia wstępnych założeń co do topologii modelu. Z tego to względu w ostatnich latach następuje znaczący wzrost zainteresowania tworzeniem nowych bądź udoskonalaniem obecnych metod budowy modelu.

1.2.2 Łańcuchy Markowa

Łańcuchy Markowa najprościej określić można za pomocą jednej z dostępnych definicji. Wszystkie rozważania dotyczyć będą funkcji dyskretnej w czasie.

Definicja 1. *Ciąg zmiennych losowych (X_t) o wartościach w przeliczalnym zbiorze S_X (przestrzeni stanów) nazywamy łańcuchem Markowa wtedy i tylko wtedy, gdy dla każdego $t \in N$ i każdego ciągu $x_1, x_1, \ldots, x_t \in S_X$ mamy*

$$P(X_t = x_t \mid X_{t-1} = x_{t-1}, \ldots, X_2 = x_2, X_1 = x_1) = \\ = P(X_t = x_t \mid X_{t-1} = x_{t-1}) \tag{1}$$

jeśli tylko

$$P(X_{t-1} = x_{t-1}, \ldots, X_2 = x_2, X_1 = x_1) > 0. \tag{2}$$

Warunek w powyższej definicji określa cały charakter łańcuchów Markowa, czyli, że ewolucja procesu zależy tylko i wyłącznie od bieżącego stanu. Zatem na charakter zmiennej X_t w zadanej chwili t ma wpływ wartość procesu w chwili $t-1$. Taki łańcuch nazywamy łańcuchem Markowa rzędu pierwszego, czyli jego stan zależy tylko od stanu poprzedniego. Istnieją także rzędy x, mówiące, że stan zależy od x-stanów poprzednich.

W wypadku, gdy prawdopodobieństwa przejścia między stanami nie zależą od momentu, w którym rozpatrywany jest proces, wtedy łańcuch określić można mianem jednorodnego w czasie. Prawdopodobieństwo takie oznacza się jako p_{ij}. Macierz kwadratowa $P = [p_{ij}]$ jednorodnego łańcucha Markowa określa się jako macierz przejść.

1.2.3 Definicja ukrytego modelu Markowa

Dla rozważań w tym rozdziale, niech:
T = długość obserwowanej sekwencji
N = liczba stanów w modelu
M = liczba obserwacji

Mając na uwadze, że ukryty model Markowa jest szczególnym przypadkiem łańcuchu Markowa, możemy teraz opisać ten układ. Przejścia między stanami opisane niech będą przy pomocy kwadratowej macierzy prawdopodobieństw $A = [a_{ij}]$, gdzie element a_{ij} jest to prawdopodobieństwo przejścia od stanu i do stanu j w następnej chwili czasowej.

$$a_{ij} = Pr(x_t = j \mid x_{t-1} = i) \quad dla \quad 1 \leq i, j \leq N$$

Spełniona jest równość:

$$\sum_{j=1}^{N} a_{ij} = 1$$

Kolejną składową systemu jest macierz Π zawierająca prawdopodobieństwa wystąpienia **i**-tego stanu na początku sekwencji stanów.

$$\Pi_i = Pr(x_0 = 1)$$

Znając obie macierze A oraz Π można obliczyć prawdopodobieństwo wygenerowania przez system sekwencji stanów $X = (x_0, x_1, \ldots, x_T)$.

$$Pr(x \mid A, \Pi) = \Pi_{x_0} a_{x_0 x_1} a_{x_1 x_2} \ldots x_{x_{T-1} x_T}$$

Tak opisana została warstwa ukryta. Model ten posiada jednakże jeszcze drugą warstwę, określaną mianami warstwy ukrytej, obserwowanej, bądź emisji. Każdemu ze stanów ukrytych przypada pewne prawdopodobieństwo b, mówiące, że w trakcie przebywania w nim, wygenerowana zostanie obserwacja O_t.

$$B = \{b_i(O_t)\}_{i=1}^{N}$$

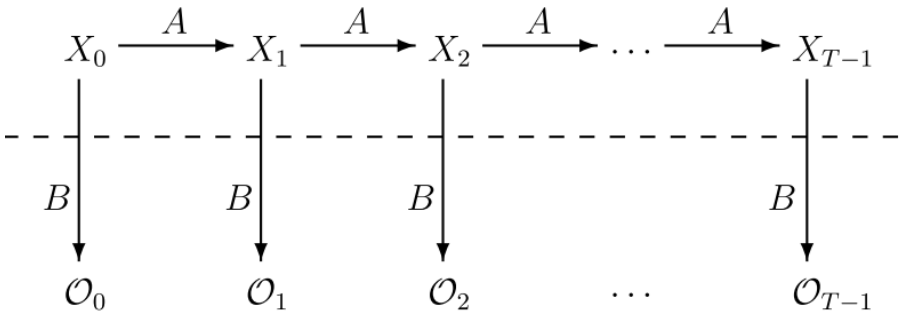
Posiadając te wszystkie informacje można określić ukryty model Markowa jako λ:

$$\lambda = (\Pi, A, B)$$

Teraz można obliczyć prawdopodobieństwo wygenerowania sekwencji obserwacji O przez system λ:

$$Pr(O \mid \Pi, A, B) = \sum_x P(O, x \mid \Pi, A, B) = \sum_x \Pi_{x_0} \prod_{t=1}^{T} a_{x_{t-1} x_t} b_{x_t}(O_t)$$

Ukryty model Markowa najprościej jest zobrazować przy pomocy poniższego schematu, zwanego grafem *Trellisa*, gdyż ukazuje zmienność procesu wraz z czasem. Linią kropkowaną oddzielono to co widoczne jest przez obserwatora od warstwy ukrytej. Strzałki między węzłami mówią o bezpośredniej zależności stochastycznej, zaś brak strzałki oznacza niezależność losową.



Rysunek 5: Ukryty model Markowa

Gdzie:

X = $(X_0, X_1, ..., X_{T-1})$ ukryte stany

O = $(O_0, O_1, ..., O_{T-1})$ sekwencja obserwacji

A = macierz prawdopodobieństw przejść między stanami

B = macierz prawdopodobieństw wystąpienia obserwacji (emisji)

2 Opis przeprowadzonych badań

2.1 Sieci Bayesowskie

Badania przeprowadzone na zaimplementowanych sieciach bayesowskich miały za zadanie określenie różnic w działaniu modelu wynikających z zastosowania różnych algorytmów konstruowania sieci.
Algorytmami wybranymi do przeprowadzenia badań były:

1. Algorytm naiwny
2. Algorytm Chow–Liu
3. Algorytm „Hill Climber"
4. Algorytm symulowanego wyżarzania
5. Algorytm „LAGD Hill Climber"

Przeprowadzenie badań zostało rozpoczęte od utworzenia sieci bayesowskich korzystających z wymienionych powyżej algorytmów w do utworzenia struktury grafu.
Następnie utworzone sieci bayesowskie poddano uczeniu za pomocą fragmentu zbioru danych dotyczącego występowania nawrotu choroby nowotworowej piersi pochodzącego z repozytorium *UCI*[1].
Wspomniany zbiór składa się z zestawu dziewięciu cech opisywanych wartościami dyskretnymi oraz dwóch klas.
Przed rozpoczęciem procesu uczenia atrybuty zbioru uczącego zostały przekonwertowane na format „One Hot Encoding" (OHE).
Po zakończeniu procesu uczenia na wejście modeli został podany fragment zbioru danych rozłączny ze zbiorem wykorzystanym do uczenia. Na podstawie danych otrzymanych w wyniku predykcji dokonywanych przez modele obliczony został ich błąd klasyfikacji. Co więcej, dla każdego algorytmu generowania sieci sporządzono macierz błędów (tablicę pomyłek).
Na podstawie opracowanej macierzy błędów wyznaczone zostały miary jakości modeli:

- prawdziwie pozytywna (ang. true positive TP)
- prawdziwie negatywna (ang. true negative TN)
- fałszywie pozytywna (ang. false positive FP), błąd I typu
- fałszywie negatywna (ang. false negative FN), błąd II typu
- czułość (ang. sensitivity) - odsetek prawdziwie pozytywnych (ang. true positive rate TPR)
- specyficzność (ang. specificity SPC) - odsetek prawdziwie negatywnych (ang. True Negative Rate TNR)
- precyzja (ang. precision)
- dokładność (ang. accuracy ACC)

Czułość interpretuje się jako zdolność modelu do prawidłowego rozpoznania nawrotu choroby tam, gdzie on występuje i została obliczone zgodnie ze wzorem:

$$TPR = TP/(TP + FN)$$

Gdzie:
TPR - czułość
TP - prawdziwie pozytywna

[1]Link do zbioru: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer

FN - fałszywie negatywna
Czułość 100% oznaczałaby, że wszystkie osoby z nawrotem choroby zostaną wyznaczone przez model.
Specyficzność wyznaczono na podstawie równania:

$$TNR = TN/(FP + TN)$$

Gdzie:
TNR - specyficzność
TN - prawdziwie negatywna
FP - fałszywie pozytywna
Specyficzność 100% oznaczałaby, że wszyscy ludzie zdrowi w wykonanym teście zostaną oznaczeni przez model jako zdrowi.

Precyzja jest określona wzorem:

$$PRE = TP/(TP + FP)$$

Gdzie:
PRE - precyzja
TP - prawdziwie pozytywna
FP - fałszywie pozytywna

Dokładność obliczono korzystając z równania:

$$ACC = (TP + TN)/(P + N)$$

Gdzie:
ACC - dokładność
TP - prawdziwie pozytywna
TN - prawdziwie negatywna
P - wartości wyznaczone poprawnie
N - wartości wyznaczone błędnie

Dodatkowo wygenerowane zostały grafy przedstawiające strukturę sieci bayesowskiej analizowanych modeli, dzięki czemu możliwe było bardziej szczegółowe przeanalizowanie wyników.

Na koniec wykonane zostały także symulacje działania klasyfikatorów:

- Sieci neuronowej z wsteczną propagacją błędów
- Sieci neuronowej typu ELM
- Support Vector Machine

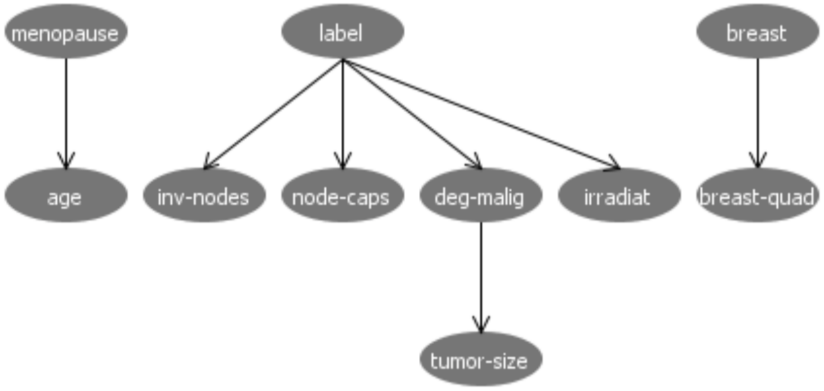
na tym samym zbiorze danych. Dzięki temu możliwe było odniesienie otrzymanych przez sieci bayesowskie wyników do rezultatów osiąganych przez inne rodzaje klasyfikatorów.

2.1.1 Sieć bayesowska otrzymana za pomocą algortymu „Hill Climber”

Tabela 1: Macierz błędu sieci bayesowskiej otrzymanej za pomocą algortymu „Hill Climber”

		Klasa przewidywana	
		pozytywna	negatywna
Klasa rzeczywista	pozytywna	4	17
	negatywna	9	28

Miary jakości otrzymanych wyników:
Poprawnie zaklasyfikowane: 32
Błędnie zaklasyfikowane: 26
Czułość: 30,77%
Specyficzność: 62,22%
Precyzja: 19,05%
Dokładność: 55,17%
Błąd: 44,83%



Rysunek 6: Graf sieci bayesowskiej otrzymanej za pomocą algortymu „Hill Climber”

Algorytm Hill Climber charakteryzuje się umiejętnością znajdowania ekstremów lokalnych. Niestety podczas procesu optymalizacji ma tendencję do znajdowania przeciętnych rozwiązań , które wydają się obiecujące jedynie lokalnie. Prawdopodobnie właśnie dlatego otrzymane przez algorytm Hill Climber rozwiązanie charakteryzuje się niską dokładnością (55% przy problemie dwuklasowym).
abstractname Warto zwrócić także uwagę na niską wartość precyzji oraz czułości. Istotną obserwacją jest fakt, że model znacznie częściej wybierał klasę negatywną, co prowadziło do zawyżenia wartości specyficzności i obniżenia czułości.
Wybór klasy negatywnej może wynikać ze zbytniego dopasowania się do zbioru uczącego, w wyniku czego prawdopodobieństwo przynależności do klasy pozytywnej wyliczane dla obiektów ze zbioru testowego było stosunkowo niskie.
Warto zauważyć, że algorytm bardzo dobrze poradził sobie ze znalezieniem prawidłowości występujących pomiędzy wiekiem pacjentki, a wiekiem wystąpienia u niej menopauzy oraz zależności

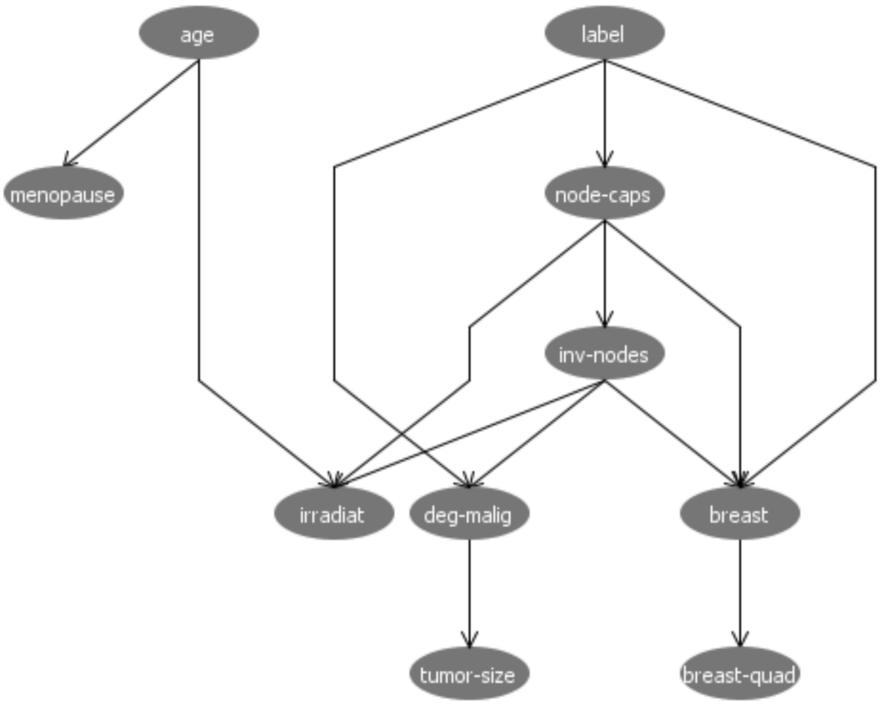
pomiędzy piersią dotkniętą nowotworem, a kwartylem piersi (w zbiorze wystąpiła duplikacja informacji). Algorytmowi nie udało się zbudować więcej niż dwóch poziomów zależności przyczynowo skutkowych, co może być jedyną z przyczyn jego słabych wyników.

2.1.2 Sieć bayesowska otrzymana za pomocą algorytmu symulowanego wyżarzania

Tabela 2: Macierz błędu sieci bayesowskiej otrzymanej za pomocą algorytmu symulowanego wyżarzania

		Klasa przewidywana	
		pozytywna	negatywna
Klasa rzeczywista	pozytywna	3	18
	negatywna	7	30

Miary jakości otrzymanych wyników:
Poprawnie zaklasyfikowane: 33
Błędnie zaklasyfikowane: 25
Czułość: 30,00%
Specyficzność: 62,50%
Precyzja: 14,29%
Dokładność: 56,90%
Błąd: 43,10%



Rysunek 7: Graf sieci bayesowskiej otrzymanej za pomocą algorytmu symulowanego wyżarzania

Algorytm symulowanego wyżarzania eliminuje podstawową wadę algorytmu Hill Climber - tendencję do utykania w ekstremum lokalnym. Algorytm poprzez wykonywanie skoków w losowe

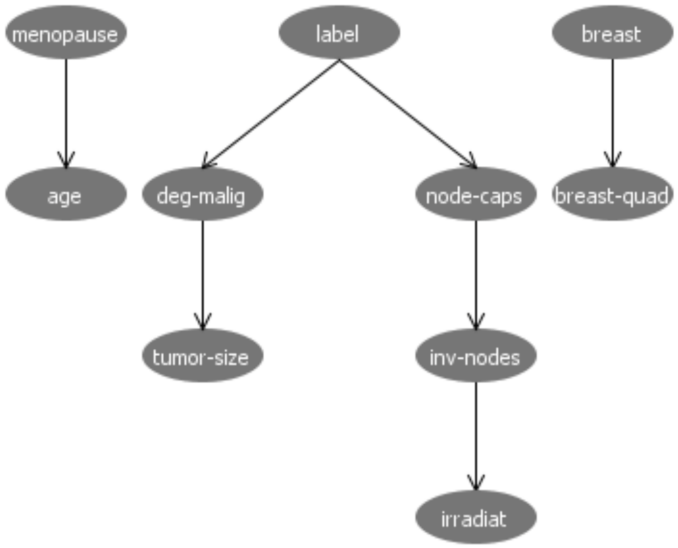
miejsca (prawdopodobieństwo określone rozkładem) w kolejnych iteracjach algorytmu. Zastosowanie bardziej skomplikowanej heurystyki zaowocowało uzyskaniem wyższej o około 1.73% dokładności modelu.
abstractname Lekkiemu obniżeniu uległa precyzja, a pozostałe metryki pozostały na zbliżonym poziomie. Graf skonstruowany przez algorytm symulowanego wyżarzania charakteryzuje się znacznie większą liczbą znalezionych zależności przyczynowo-skutkowych niż w przypadku poprzedniego algorytmu.
Dodatkowo graf zawiera znacznie mniej liści, co może sugerować, że niektóre zależności mogą być nieprawidłowe - zgodnie z grafem dwa stany są pośrednio lub bezpośrednio zależne od pięciu innych. Być może ta właściwość grafu doprowadziła do niskiej skuteczności klasyfikatora.

2.1.3 Sieć bayesowska otrzymana za pomocą algortymu „LAGD Hill Climber"

Tabela 3: Macierz błędu sieci bayesowskiej otrzymanej za pomocą algorytmu „LAGD Hill Climber"

		Klasa przewidywana	
		pozytywna	negatywna
Klasa rzeczywista	pozytywna	3	18
	negatywna	3	34

Miary jakości otrzymanych wyników:
Poprawnie zaklasyfikowane: 37
Błędnie zaklasyfikowane: 21
Czułość: 50,00%
Specyficzność: 65,38%
Precyzja: 14,29%
Dokładność: 63,79%
Błąd: 36,21%



Rysunek 8: Graf sieci bayesowskiej otrzymanej za pomocą algorytmu „LAGD Hill Climber"

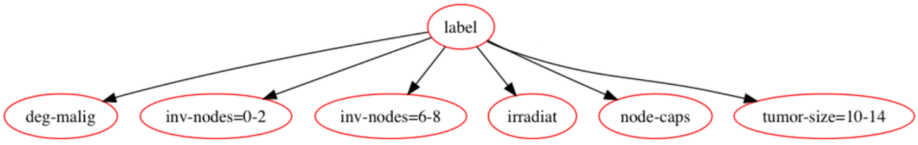
Algorytm LAGD Hill Climber stanowi rozszerzenie algorytmu Hill Climber polegające na analizie dodatkowo kilku kroków algorytmu wprzód w celu odnalezienia najlepszej struktury grafu. Dzięki zastosowaniu wspomnianego usprawnienia udało się otrzymać dokładność klasyfikacji wyższą o około 8,5%, co stanowi świetny rezultat.
Powstały graf ma podobną strukturę do grafu powstałego w efekcie działania Hill Climber. Zmianie uległo największe drzewo grafu, a pozostałe dwa mniejsze pozostały bez zmian.
Nowa struktura największego drzewa posiada dwukrotnie mniej liści, co sugeruje, że udało się odszukać głębsze zależności między stanami. Otrzymany graf stanowi formę pośrednią pomiędzy poprzednimi dwoma strukturami.
Otrzymany model w porównaniu do poprzedników charakteryzuje się wysoką czułością, ponieważ lepiej radzi sobie z prawidłowym rozpoznaniem nawrotu choroby.

2.1.4 Sieć bayesowska otrzymana za pomocą algorytmu naiwnego

Tabela 4: Macierz błędu sieci bayesowskiej otrzymanej za pomocą algorytmu naiwnego

		Klasa przewidywana	
		pozytywna	negatywna
Klasa rzeczywista	pozytywna	0	18
	negatywna	0	40

Miary jakości otrzymanych wyników:
Poprawnie zaklasyfikowane: 40
Błędnie zaklasyfikowane: 18
Czułość: niemożliwa do wyznaczenia
Specyficzność: 68,97%
Precyzja: 0,00%
Dokładność: 68,97%
Błąd: 31,03%



Rysunek 9: Graf sieci bayesowskiej otrzymanej za pomocą algorytmu naiwnego

Sieć bayesowska skonstruowana za pomocą algorytmu naiwnego składa się drzewa (korzeń i liście). Ze względu na to, że algorytmowi naiwnemu nie udało się odszukać zbyt wielu zależności przyczynowo skutkowych w zbiorze model posiada znikomą wartość jako klasyfikator.
W ramach testów przeprowadzonym na zbiorze testowym wszystkie otrzymane odpowiedzi były takie same - 'no-reccurence-events'. W związku z tym precyzja modelu wyniosła 0%. Wysoka w porównaniu z poprzedniki algorytmami skuteczności jest jedynie dziełem przypadku i wynika z

proporcji klas w zbiorze testowym. Wnioskiem z przeprowadzonego eksperymentu jest fakt, że algorytm naiwny może okazać się zbyt prosty aby odpowiednio odwzorować zależności występujące w zbiorze w postaci grafu.

2.1.5 Sieć bayesowska otrzymana za pomocą algortymu Chow-Liu

Tabela 5: Macierz błędu sieci bayesowskiej otrzymanej za pomocą algortymu Chow-Liu

		Klasa przewidywana	
		pozytywna	negatywna
Klasa rzeczywista	pozytywna	0	15
	negatywna	1	42

Miary jakości otrzymanych wyników:
Poprawnie zaklasyfikowane: 42
Błędnie zaklasyfikowane: 16
Czułość: 0,00%
Specyficzność: 73,68%
Precyzja: 0,00%
Dokładność: 72,41%
Błąd: 27,59%

Rysunek 10: Graf sieci bayesowskiej otrzymanej za pomocą algortymu Chow-Liu

Cechą algorytmu Chow-Liu jest fakt, że produkuje on niezorientowane drzewo zależności. Jeżeli sieć Bayesa danego rozkładu ma postać drzewa, to algorytm Chow-Liu poprawnie odtworzy jego kształt. Algorytm Chow-Liu nie sprawdza, czy rozkład zmiennych zadanej próby spełnia powyższy warunek.
Jeśli struktura zależności między zmiennymi w próbie nie jest drzewem, to algorytm znajduje najlepszą strukturę drzewiastą, która opisuje postać zależności. Należy pamiętać, że w niekorzystnych

wypadkach informacja zwrócona przez algorytm może być błędna.
Tak właśnie stało się w przypadku analizowanego przez nas zbioru danych. Już po samej strukturze grafu widać, że znalezione przez algorytm zależności przyczynowo-skutkowe są niepoprawne. Przykładem niepoprawnej zależności może być występowanie 6-8 ognisk nowotworu w przypadku występowania 0-2 ognisk nowotworu (wystąpienie tych sytuacji wzajemnie się wyklucza).
W efekcie otrzymany model nie nadaje się do wykorzystania w celach diagnostycznych, a wartości parametrów czułości i precyzji wynoszą 0% pomimo wysokiej dokładności (72%). Podobnie jak w przypadku algorytmu naiwnego, dokładność odwzorowała jedynie przybliżenie rozkładu klas w zbiorze uczącym.

Tabela 6: Zestawienie dokładności klasyfikacji otrzymanej przez wszystkie modele uczestniczące w przeprowadzonym eksperymencie

Model	Dokładność klasyfikacji
Sieć bayesowska (algorytm naiwny)	68.97%
Sieć bayesowska (Chow–Liu)	72.41%
Sieć bayesowska („Hill Climber")	55,17%
Sieć bayesowska (symulowane wyżarzanie)	56,90%
Sieć bayesowska („„LAGD Hill Climber")	63,79%
Sieć neuronowa z wsteczną propagacją	71,08%
Sieć neuronowa typu ELM	69,97%
Support Vector Machine	73,11%

Eksperymenty przeprowadzone na zestawie ośmiu modeli pokazują, że efekty uzyskane przez sieci bayesowskie są wyraźnie gorsze od bardziej skomplikowanych modeli. Wyniki uzyskane przez algorytm naiwny oraz Chow-Liu zostały zignorowane ze względu na bardzo niskie miary jakości tych modeli. Spośród sieci bayesowskich najlepiej poradził sobie algorytm LAGD Hill Climber. Dzięki przewidywaniu na kilka kroków wprzód, która stryktura grafu może okazać się skuteczniejsza udało mu się poprawić wyniki klasycznego algorytmu Hill Climber o ponad 8% c stanowi znaczącą różnicę. Algorytm symulowanego wyżarzania osiągnął poziom podobny do klasycznego algorytmu Hill Climber pokonując go o około 1,7%. Sieć typu ELM ze względu na swoją uproszczoną strukturę uplasowała się poniżej sieci neuronowej ze wsteczną propagacją (jednak udało się jej uzyskać znacznie lepsze czasu uczenia). Najlepszym klasyfikatorem okazał się SVM, któremu udało się osiągnąć rezultat lepszy o 10% niż LAGD Hill Climber oraz około 18% lepszy od pozostałych sieci bayesowskich.

2.2 Ukryte modele Markowa

Badania dla ukrytego modelu Markowa przeprowadzono dla zobrazowania podstawowych zależności zachodzących między zbiorem podstawowym nie posiadającym żadnej wiedzy o problemie, czy w tym wypadku chorobie, oraz zbioru sekwencyjnego, taką wiedzę posiadającą. Ideą zbioru sekwencyjnego było aby składał się on z danych wskazujących w jaki sposób zmieniała się choroba wraz z czasem.

W projekcie wykorzystano zbiór charakteryzujący problem raka piersi z darmowego internetowego zasobu *UCI*, który to nie posiadał historii choroby[2]. Należało zatem historię przemian wygenerować własnoręcznie. Posłużono się w tym celu macierzą przejść łańcucha Markowa, który przedstawiał prawdopodobieństwa przejścia od klasy X do klasy Y, bądź zostania dalej w klasie X. Cały schemat generowania sekwencji przedstawić można w kilku krokach:

1. Stwórz macierz przejść łańcucha Markowa
2. Posortuj próbki po klasie
3. Dla każdej próbki:
   (a) Wybierz następną klasę z prawdopodobieństwem z macierzy
   (b) Dla wybranej klasy wybierz próbkę z prawdopodobieństwem losowym
   (c) Dopisz wybraną klasę i próbkę jako sekwencja

Posiadając oba zbioru - bazowy oraz sekwencyjny, przeprowadzono badania najważniejszych parametrów modelu - czasu uczenia, czasu testowania oraz efektywności. Manipulowano przy tym dostępnym parametrem - algorytmem dekodowania, gdzie starano się zbadać jaki wpływ na wyniki ma zmiana algorytmu. Zastosowano dwa algorytmy - algorytm Viterbiego bazujący na technice programowania dynamicznego, oraz algorytm heurystyczny "best-first" znany także jako algorytm dekodowania posteriori.

Jako stałe parametry przyjęto długość sekwencji na 2, gdyż wstępne badania wykazały, iż zwiększenie sekwencji nie wpływa w zauważalny sposób na wyniki. Niezmienny był także współczynnik zbioru testowego, wynoszący 0.2, co wskazywało, iż 20% zbioru wejściowego wykorzystywane jest jako zbiór testowy. Następnym niezmiennym parametrem był parametr wejściowy do algorytmu - alfa, który to w literaturze angielskiej określany jest jako *Lidstone (additive) smoothing parameter*, a jego zmiana również nie wpływała w zauważalny sposób na wyniki badań.

W celu oszczędzenie miejsca na poniższych wykresach skrócono nazewnictwo, które oznacza:

- *nonseq Viterbi* - Badanie przy użyciu zbioru bez sekwencji oraz algorytmu Viterbiego
- *seq Viterbi* - Badanie przy użyciu zbioru z sekwencjami oraz algorytmu Viterbiego
- *nonseq bestfirst* - Badanie przy użyciu zbioru bez sekwencji oraz algorytmu "best-first"
- *seq bestfirst* - Badanie przy użyciu zbioru z sekwencjami oraz algorytmu "best-first"

Wszystkie badania powtórzono 100 razy i na podstawie wyników wyciągnięto podstawowe wartości statystyczne jak:

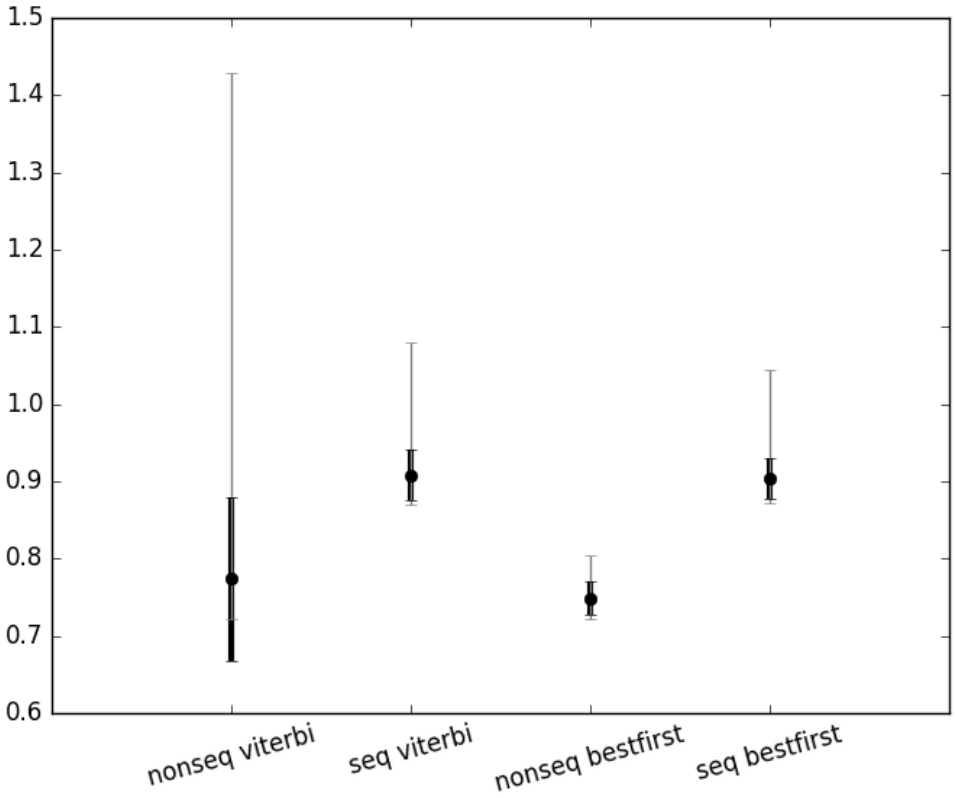
- *max* - wartość minimalna
- *min* - wartość maksymalna
- *mean* - wartość średnia
- *std* - odchylenie standardowe

[2]Link do zbioru: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer

Pierwszymi badaniami było sprawdzenie czasu uczenia modelu. Jak widać, czas uczenia nieznacząco rośnie o około 15% jeśli chodzi o zbiór sekwencyjny względem zbioru bazowego i jest to widoczne dla obu algorytmów. Dwa algorytmy dają bardzo zbliżone do siebie rezultaty, jednakże odchylenie standardowe i różnica między maksymalną osiągniętą wartością a minimalną są mniejsze w wypadku algorytmu *best-first*, co sugeruje, że zapewnia on stabilniejsze czasy nauczania.

Tabela 7: Wyniki badań - czas uczenia (w ms)

	nonseq Viterbi	**seq Viterbi**	**nonseq best-first**	**seq best-first**
min	0.7223	0.8702	0.7221	0.8716
max	1.4279	1.0790	0.8037	1.0435
mean	0.7734	0.9079	0.7487	0.9040
std	0.1063	0.0329	0.0209	0.0264

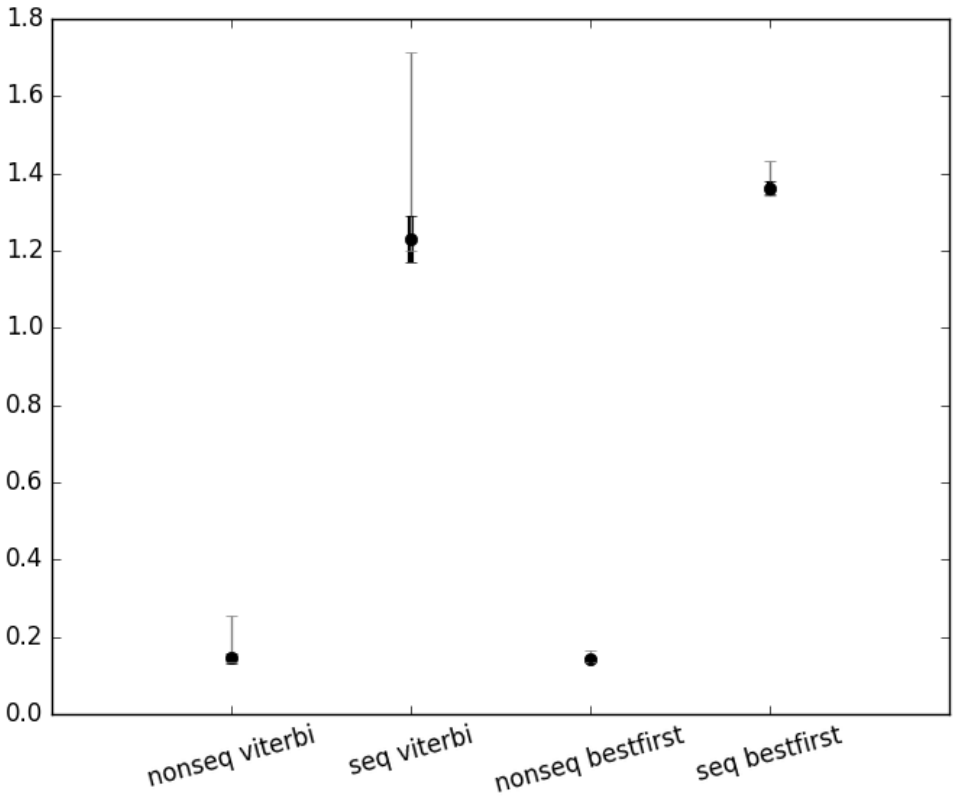


Rysunek 11: Czas uczenia (w ms)

Kolejnym badaniem było sprawdzenie czasu testowania. Tutaj już widać znaczący, bo aż blisko 10-krotny wzrost czasu testowania zbioru sekwencyjnego względem zbioru bazowego. Różnice między algorytmami również nie są bardzo widoczne, zauważyć jednakże również można mniejszy rozrzut algorytmu *best-first*.

Tabela 8: Wyniki badań - czas testowania (w ms)

	nonseq Viterbi	**seq Viterbi**	**nonseq best-first**	**seq best-first**
min	0.1368	1.1994	0.1366	1.3415
max	0.2546	1.7130	0.1640	1.4338
mean	0.1445	1.2293	0.1418	1.3628
std	0.0133	0.0607	0.0050	0.0157

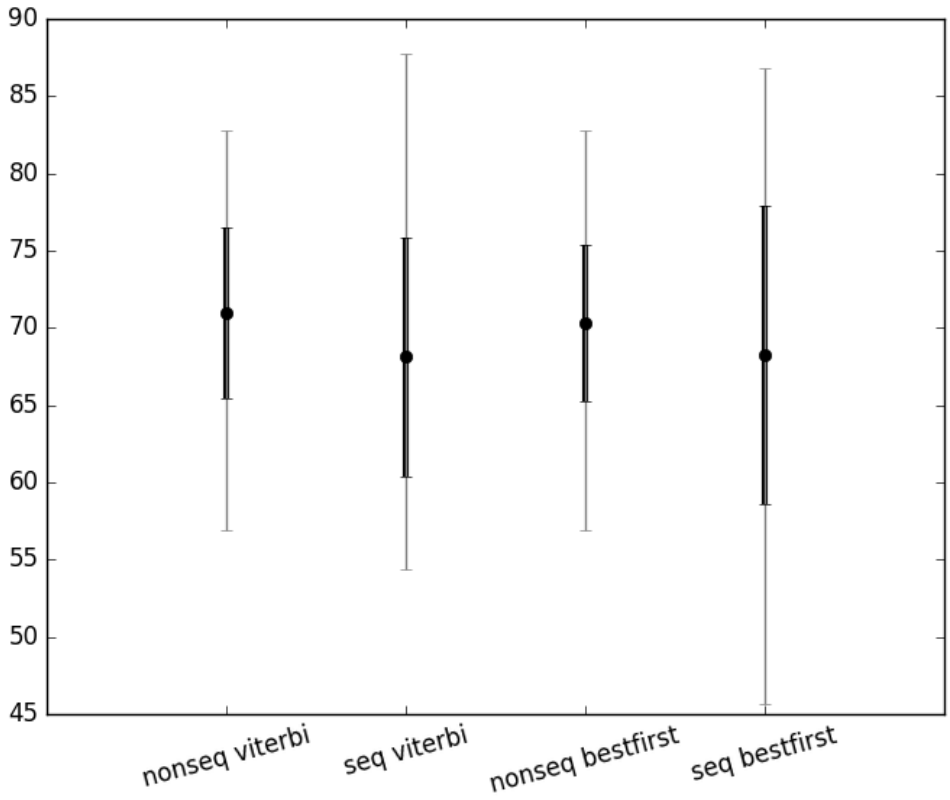


Rysunek 12: Czas testowania (w ms)

Ostatnim przeprowadzonym eksperymentem było sprawdzenie efektywności modelu. Zauważyć można, iż na efektywność nie ma wpływu rodzaj zastosowanego zbioru, a wręcz zauważyć można, że dla zbioru sekwencyjnego rośnie rozrzut wartości względem zbioru bazowego, co sugeruje jego mniejszą stabilność. Różnice między dwoma algorytmami są także prawie niezauważalne.

Tabela 9: Wyniki badań - dokładność (w %)

	nonseq Viterbi	**seq Viterbi**	**nonseq best-first**	**seq best-first**
min	56.8965	54.3859	56.8965	45.6140
max	82.7586	87.7192	82.7586	86.8421
mean	70.9827	68.1140	70.2931	68.2456
std	5.5390	7.7356	5.0435	9.6267



Rysunek 13: Dokładność (w %)