

Wydział Podstawowych Problemów Techniki

**PRACA DYPLOMOWA**

**Tytuł pracy dyplomowej:**

**MASZYNOWE UCZENIE GRAMATYCZNYCH DESKRYPTORÓW SEKWENCJI BIAŁKOWYCH**

**Autor: Robert Kowalski**

**Opiekun: dr inż. Witold Dyrka**

słowa kluczowe: aminokwasy, algorytmy ewolucyjne, teoria automatów, język formalny, gramatyka, probabilistyczna gramatyka bezkontekstowa, analizator składniowy

krótkie streszczenie: Celem niniejszej pracy jest wprowadzenie ulepszeń do istniejącej metody probabilistycznego modelu gramatycznego sekwencji białkowych w zakresie maszynowego uczenia modelu. W pracy opisano wykorzystywaną w algorytmach pierwszorzędową strukturę białek, obecnie wykorzystywane w bioinformatyce algorytmy ewolucyjne (z uwzględnieniem ich mocnych i słabych stron), procesy uczenia maszynowego, podstawowe pojęcia związane z teorią automatów (takie jak alfabet, łańcuch, język oraz problem w kontekście teorii automatów) oraz języki i gramatyki ze szczególnym uwzględnieniem probabilistycznych gramatyk bezkontekstowych.

# SPIS TREŚCI

[SPIS TREŚCI 2](#_Toc534260524)

[WPROWADZENIE 3](#_Toc534260525)

[1.ALGORYTMY EWOLUCYJNE 3](#_Toc534260526)

[1.1. WPROWADZENIE DO ALGORYTRMÓW EWOLUCYJNYCH 3](#_Toc534260527)

[1.2. DZIAŁANIE ALGORYTMÓW EWOLUCYJNYCH 3](#_Toc534260528)

[1.3. PROSTY ALGORYTM GENETYCZNY 4](#_Toc534260529)

[1.3.1. Właściwości prostego algorytmu genetycznego 4](#_Toc534260530)

[1.3.2. Kodowanie osobników i operacje genetyczne w algorytmie SGA 5](#_Toc534260531)

[1.3.3. Słabości algorytmów ewolucyjnych 6](#_Toc534260532)

[1.4. GRAMATYKI 7](#_Toc534260533)

[1.4.1. Podstawowe pojęcia związane z gramatykami 7](#_Toc534260534)

[1.4.2. Gramatyki formalne 8](#_Toc534260535)

[1.4.3. Probabilistyczne gramatyki bezkontekstowe 9](#_Toc534260536)

[1.5. ANALIZATORY SKŁADNIOWE 9](#_Toc534260537)

[2.BADANE JĘZYKI 10](#_Toc534260538)

[2.1. PIERWSZY JĘZYK TESTOWY 10](#_Toc534260539)

[2.2. DRUGI JĘZYK TESTOWY 11](#_Toc534260540)

[2.3. JĘZYK OPISUJĄCY SEKWENCJE NALEŻĄCE DO RODZINY MOTYWÓW BIAŁKOWYCH 12](#_Toc534260541)

[3.ZASTOSOWANY ALGORYTM MASZYNOWEGO UCZENIA 13](#_Toc534260542)

[4.PRZEBIEG EKSPERYMENTU 15](#_Toc534260543)

[4.1. WPŁYW HIPERPARAMETRÓW ALGORYTMU MASZYNOWEGO UCZENIA NA PRZEBIEG I WYNIKI NAUKI 15](#_Toc534260544)

[4.1.1. Wpływ liczby osobników na przebieg i wyniki nauki 17](#_Toc534260545)

[4.1.2. Wpływ prawdopodobieństwa mutacji na przebieg i wyniki nauki 18](#_Toc534260546)

[4.1.3. Wpływ maksymalnej skali mutacji na przebieg i wyniki nauki 20](#_Toc534260547)

[4.1.4. Wpływ prawdopodobieństwo krzyżowania na przebieg i wyniki nauki 22](#_Toc534260548)

[4.2. WPŁYW LICZBY SYMBOLI NIETERMINALNYCH NA WYNIKI NAUKI 23](#_Toc534260549)

[4.3. TESTOWANIE MASZYNOWEGO UCZENIA NA JĘZYKU OBEJMUJĄCYM SEKWENCJE AMINOKWASOWE RODZINY MOTYWÓW BIAŁKOWYCH 24](#_Toc534260550)

[5.PODSUMOWANIE 28](#_Toc534260551)

[LITERATURA 29](#_Toc534260552)

# WPROWADZENIE

Struktura pierwszorzędowa białek, to najbardziej podstawowy i najmniej skomplikowany poziom organizacji strukturalnej białka, opisujący w formie liniowej układ aminokwasów tworzących rozpatrywany łańcuch polipeptydowy.

Fizykochemiczne właściwości białek są zależne od ich struktury która jest warunkowana przez sekwencję aminokwasów tworzących dane białko, przy czym, podobnie jak w przypadku słów w języku naturalnym którego używamy na co dzień do komunikacji interpersonalnej, sekwencje aminokwasów mogą być niejednoznaczne (jedna sekwencja aminokwasów może odpowiadać za więcej niż jedną strukturę białka w zależności od kontekstu (środowiska w którym się znajduje)). Można zauważyć, że te kluczowe dla wszelkiego życia struktury wykazują podstawową charakterystykę języków: praktycznie nieskończona liczba sekwencji może zostać wyrażona za pomocą skończonej liczby monomerów. W przypadku białek zaledwie 20 niepowtarzalnych aminokwasów (które mogą być traktowane jako litery) tworzy miliony sekwencji (które mogą być traktowane jako pojedyncze słowa lub całe zdania) [1].

Ze względu na to, algorytmy wykorzystujące założenia koncepcji języka białek mogą być wykorzystywane jako narzędzia bioinformatyczne umożliwiające np., z dużym prawdopodobieństwem, stwierdzenie, czy dane białko należy do określonej rodziny białek, na podstawie jego struktury pierwszorzędowej (sekwencji aminokwasów tworzących dane białko) [2].

# ALGORYTMY EWOLUCYJNE

## WPROWADZENIE DO ALGORYTRMÓW EWOLUCYJNYCH

Algorytmy ewolucyjne służą do przeszukiwania przestrzeni alternatywnych rozwiązań w celu wyszukania rozwiązania optymalnego, lub rozwiązania które będzie do niego w wystarczającym stopniu zbliżone w określonym, rozsądnym czasie [3].

Terminologia wykorzystywana przy opisywaniu algorytmów ewolucyjnych jest inspirowana terminologią wykorzystywaną do opisywania procesów ewolucyjnych. Tak więc: zbiór rozwiązań rozpatrywanego problemu nazywamy populacją, która to natomiast składa się z pojedynczych osobników (będących pojedynczymi rozwiązaniami problemu). Rozwiązywany problem określany jest mianem środowiska. W zależności od jakości danego rozwiązania, każdemu osobnikowi przypisywana jest wartość liczbowa która określana jest mianem przystosowania. Każdy osobnik określany jest przez zestaw cech określany jako fenotyp, który z kolei jest kodowany przez zestaw parametrów nazywany genotypem. Dla rozpatrywanego środowiska można określić funkcję przystosowania, która dla każdego osobnika przypisuje konkretną wartość przystosowania w zależności od opisującego danego osobnika fenotypu. Genotyp osobnika składa się z chromosomów (lub pojedynczego chromosomu) które składają się z elementarnych jednostek nazywanych genami [4].

## DZIAŁANIE ALGORYTMÓW EWOLUCYJNYCH

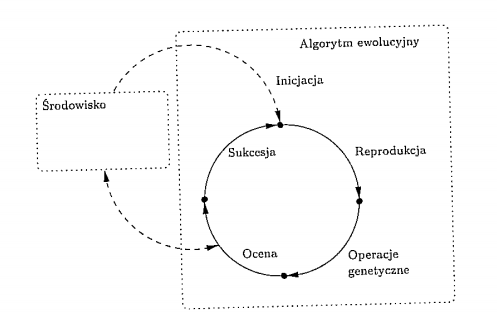
Działanie algorytmu ewolucyjnego (zilustrowane na Rysunek 1.1.1) polega na kolejnym przeprowadzaniu procesów reprodukcji, operacji genetycznych, oceny oraz sukcesji.

Proces reprodukcji polega na losowym powielaniu poszczególnych osobników składających się na rozpatrywaną populację (przy czym prawdopodobieństwo powielenia konkretnego osobnika jest tym większe im lepsze jest jego przystosowanie do rozpatrywanego środowiska). Podczas procesu reprodukcji każdy osobnik może zostać powielony zarówno kilka razy, jak i ani razu.

Powstały w wyniku reprodukcji zbiór osobników jest nazywany osobnikami rodzicielskimi, są one poddawane operacjom genetycznym takim jak mutacje, które polegają na losowym modyfikowaniu opisującego danego osobnika genotypu (często przyjmuje się, że niewielkie mutacje są bardziej prawdopodobne niż duże). Krzyżowanie to operacja polegająca na wygenerowaniu osobnika potomnego którego chromosomy są efektem wymieszania odpowiednich chromosomów osobników rodzicielskich (co najmniej dwóch), uzyskana w ten sposób grupa osobników nazywana jest populacją potomną.

W etapie oceny środowiska każdemu osobnikowi z populacji potomnej przypisywana jest konkretna wartość przystosowania (na podstawie funkcji przystosowania).

W etapie sukcesji tworzona jest nowa populacja bazowa (w skład której mogą wchodzić osobniki wyłącznie z populacji potomnej, jak i osobniki z populacji potomnej wraz z osobnikami z poprzedniej populacji bazowej) [4].



Rysunek 1.1-schemat działania algorytmu ewolucyjnego [4]

## PROSTY ALGORYTM GENETYCZNY

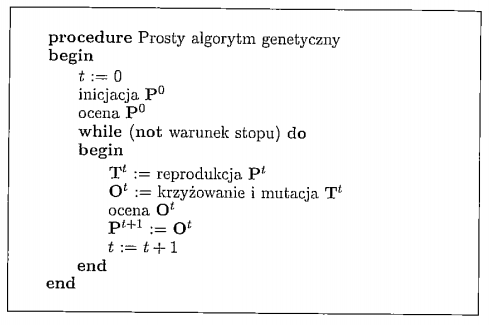
### Właściwości prostego algorytmu genetycznego

Jako prosty algorytm genetyczny (ang. Simple Genetic Algorithm, SGA) określa się zaproponowany w 1975 roku przez Johna Hollanda algorytm mający początkowo za zadanie modelowanie procesu ewolucji. Jego działanie zostało zilustrowane na Rysunek 1.2.

Algorytm przetwarza dwie populacje: Pt, którą nazywamy populacją bazową, oraz Ot, czyli populację potomną. W ramach działania algorytmu tworzona jest też populacja Tt, zwana populacją tymczasową, której zadaniem jest przechowywanie kopii osobników z populacji bazowej. Rozmiary tych trzech populacji są jednakowe.

W kroku inicjacji populacji bazowej, dochodzi do losowej generacji osobników tworzących pierwszą populację bazową, następnie obliczana jest wartość przystosowania osobników.

Osobniki z populacji bazowej są powielane z prawdopodobieństwem proporcjonalnym do ich wartości przystosowania, następnie osobniki z populacji tymczasowej są kojarzone w pary, dla każdej pary podejmuje się decyzję o krzyżowaniu (parametr określający prawdopodobieństwo decyzji pozytywnej jest atrybutem wejściowym funkcji korzystającej z algorytmu SGA). W przypadku decyzji pozytywnej osobniki z pary mieszają swoje geny, powstałe w ten sposób osobniki potomne zastępują swoich rodziców. Następnie wszystkie osobniki z populacji tymczasowej są poddawane mutacjom,  
w ten sposób powstaje populacja potomna, która w następnym obiegu pętli staje się populacją bazową. Algorytm ten wykonuje się w pętli, której każdy obieg jest nazywany generacją, pętla ta może zostać przerwana po wykonaniu określonej liczby generacji, bądź wygenerowanie osobnika o określonej wartości przystosowania.[4]



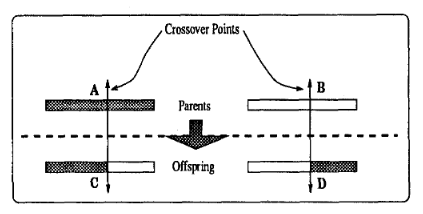
Rysunek 1.2-pseudokod algorytmu SGA[4]

### Kodowanie osobników i operacje genetyczne w algorytmie SGA

Najprostszym kodowaniem osobników stosowanym w algorytmach genetycznych znany jest z kodowanie binarne, zgodnie z którym genotyp każdego osobnika wchodzącego w skład populacji składa się z wektorów wypełnionych 0 lub 1 (wektory  
te pełnią funkcję chromosomów, zaś każdy element wektora jest pojedynczym genem).

Podczas mutacji każdy gen w genotypie danego osobnika jest rozpatrywany niezależnie, jeżeli dla konkretnego genu zostanie wylosowana decyzja o mutacji (prawdopodobieństwo takiego zdarzenia jest atrybutem wejściowym funkcji wykorzystującej algorytm SGA), to wartość genu zostaje zanegowana.

Podczas procesu krzyżowania (zilustrowanego na Rysunku 1.5) losowane jest konkretne miejsce na chromosomie (z rozkładu równomiernego), następnie chromosomy obu krzyżowanych osobników rozcinane są w wylosowanym miejscu i pierwszy fragment chromosomu osobnika pierwszego łączy się z drugim fragmentem chromosomu osobnika drugiego, zaś drugi fragment chromosomu osobnika pierwszego łączy się  
z pierwszym fragmentem chromosomu osobnika drugiego.[5]



Rysunek 1.3-Schemat procesu krzyżowania [5]

Prawdopodobieństwo powielenia konkretnego osobnika z populacji bazowej w fazie reprodukcji wyraża się wzorem:

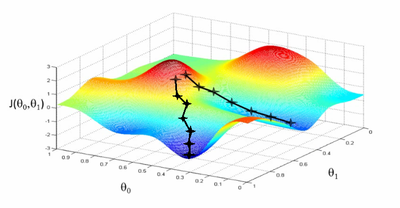
|  |  |  |
| --- | --- | --- |
|  |  |  |

Gdzie *X* oznacza osobnika, *Φ(X)*oznacza wartość przystosowania. Metoda ta jest nazwana reprodukcją proporcjonalną, bądź reprodukcją ruletkową [4].

Najczęściej stosowane w praktyce warunki przerwania algorytmu zostały opisane już wcześniej, należy jednak mieć na uwadze to, że algorytm genetyczny jest procesem losowym, w związku z czym nigdy nie jesteśmy w stanie uzyskać gwarancji, że otrzymany wynik jest wynikiem optymalnym, wiemy jedynie, że wraz ze wzrostem liczby generacji rośnie prawdopodobieństwo zbliżenia się do wyniku optymalnego [3].

### Słabości algorytmów ewolucyjnych

Ze względu na uwzględnienie wartości przystosowania podczas procesu reprodukcji, geny osobników wykazujących niskie przystosowanie do rozpatrywanego środowiska, zostaną z dużym prawdopodobieństwem wykluczone z populacji bazowej po pewnej ilości obiegów pętli algorytmu, natomiast wszelkie mutacje powodujące zmniejszenie przystosowania zostaną najprawdopodobniej odrzucone (podczas gdy mutacje pozwalające poprawić wartość przystosowania do środowiska zostaną zachowane i powielone w kolejnych populacjach bazowych). Ze względu na to, zachodzi wysokie prawdopodobieństwo, że wraz  
ze wzrostem liczby obiegów pętli algorytmu będzie rosnąć poziom przystosowania kolejnych populacji bazowych. Kierunek rozwoju kolejnych populacji bazowych jest jednak mocno uzależniony  
od przebiegu funkcji przystosowania, parametrów pierwszej populacji bazowej oraz od rodzaju zachodzących mutacji, jeżeli kolejna populacja bazowa znajdzie się w lokalnym maksimum funkcji przystosowania to każda kolejna zmiana fenotypu będzie powodowała zmniejszenie przystosowania tej populacji do danego środowiska, w konsekwencji uniemożliwiając opuszczenie lokalnego maksimum funkcji przystosowania (które może być o wiele mniejsze od globalnego maksimum tejże funkcji, zostało to zilustrowane na Rysunek 1.4). Algorytmy genetyczne są więc, jak wiele innych metod optymalizacji, podatne na osiadanie w maksimach lokalnych. Istnieją jednak metody umożliwiające zwiększenie szansy na opuszczenie takich pułapek ewolucyjnych, metodą taką jest np. odpowiednie sterowanie skalą i prawdopodobieństwem mutacji, oraz zwiększanie prawdopodobieństwa reprodukcji osobników słabiej przystosowanych.



Rysunek 1.4 - przedstawienie prawdopodobnego procesu zwiększania przystosowania populacji bazowej wraz z kolejnymi iteracjami algorytmu ewolucyjnego[6]

## GRAMATYKI

### Podstawowe pojęcia związane z gramatykami

Podstawowe pojęcia związane z gramatykami to: alfabety, łańcuchy oraz języki.

Alfabetem nazywamy konkretny, skończony, niepsuty zbiór symboli, przykładami alfabetu mogą być:

* zbiór wszystkich bądź niektórych liter alfabetu łacińskiego,
* zbiór symboli ASCII,
* zbiór złożony z dokładnie jednego symbolu,
* zbiór liczb należących do pewnego przedziału,
* stosowany w informatyce alfabet binarny,
* zbiór 20 symboli przypisanych do konkretnych aminokwasów (stosowany  
  w bioinformatyce).

Alfabety zwyczajowo oznaczamy symbolem ∑.

Łańcuchem nazywamy skończony zbiór symboli należących do konkretnego alfabetu, w przypadku w którym znaki danego łańcucha należą do konkretnego alfabetu mówimy, że jest to łańcuch nad tym alfabetem (np. 010010 to łańcuch nad alfabetem binarnym). W przypadku łańcuchów możemy wyróżnić łańcuch pusty, czyli łańcuch o zerowej liczbie wystąpień symboli, łańcuch ten oznaczamy symbolem *ε* i jest on łańcuchem nad dowolnym alfabetem.

Językiem nazywamy zbiór łańcuchów, których wszystkie elementy zostały wybrane z określonego zbioru wszystkich łańcuchów nad danym alfabetem. Jeżeli dany język jest opisywany przez konkretny alfabet, to mówimy, że jest to język nad tym alfabetem. Język nie musi wykorzystywać wszystkich symboli zawartych w danym alfabecie, w związku z czym możemy stwierdzić, że jeżeli dany język jest językiem nad alfabetem **∑**, to jest on również językiem nad dowolnym alfabetem który jest nadzbiorem alfabetu **∑** [7].

### Gramatyki formalne

Gramatyki są sposobem opisu języków formalnych (czyli podzbiorów wszystkich słów nad danym alfabetem).

Najpopularniejszym sposobem aby opisać język formalny jest użycie gramatyki formalnej, gramatyka formalna składa się z:

* Symboli terminalnych (zbiór symboli alfabetu na którym zbudowany jest język),
* Symboli nieterminalnych (dowolny skończony zbiór składający się z symboli pomocniczych),
* Symbolu startowego (który należy do symboli nieterminalnych),
* Skończonego zbioru reguł przepisywania (produkcji), każda produkcja to para słów (gdzie pierwsze słowo, nazywane poprzednikiem, w ramach konkretnej gramatyki może przechodzić w drugie), w których mogą występować zarówno symbole nieterminalne jak i terminalne.

Aby sprawdzić, czy konkretne słowo należy do języka opisanego przez taką gramatykę należy sprawdzić, czy możliwe jest wyprowadzenie tego słowa od symbolu startowego za pomocą zawartych w danej gramatyce produkcji.

Przykładowo, jeżeli rozpatrywany alfabet to alfabet binarny składający się z symboli 0 i 1, natomiast opisująca go gramatyka zawiera symbole pomocnicze A, B i C, gdzie A jest symbolem startowym, natomiast zbiór produkcji tej gramatyki to: (A→BC), (B→10) oraz (C→1) to możemy stwierdzić, że słowo: „101” należy do tej gramatyki ponieważ możliwe są takie przekształcenia symbolu startowego, że uzyskamy sprawdzane słowo (A→BC, BC→10C, 10C→101).

### Probabilistyczne gramatyki bezkontekstowe

Gramatyki formalne, w zależności od postaci jaką przyjmują należące do niej reguły można podzielić na różne klasy. Przykładem takiej klasy gramatyk są gramatyki bezkontekstowe, w których wszystkie produkcje przyjmują postać (A→*Γ*), gdzie A to dowolny symbol nieterminalny, zaś *Γ* jest dowolnym zbiorem symboli nieterminalnych i terminalnych.

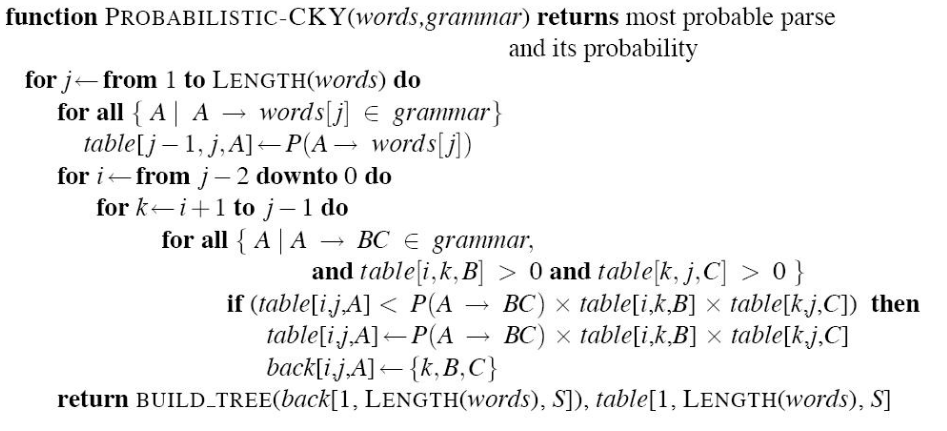
Probabilistyczne gramatyki bezkontekstowe to gramatyki bezkontekstowe, w których każdej produkcji przypisano odpowiadającą jej wartość prawdopodobieństwa wystąpienia danej produkcji. Prawdopodobieństwa dla konkretnych produkcji są przypisywane w ten sposób, aby suma prawdopodobieństw wszystkich produkcji o wspólnym poprzedniku wynosiła 1.

## ANALIZATORY SKŁADNIOWE

Analizatory składniowe, nazywane również parserami, to programy służące do analizy składniowej konkretnych danych wejściowych i określenia ich struktury gramatycznej.

Aby sprawdzić, czy konkretne słowo należy do danej gramatyki bezkontekstowej niegenerującej symbolu pustego można użyć parsera korzystającego z algorytmu Cocke'a-Youngera-Kasamiego (algorytm CKY lub CYK).[8] Schemat działania algorytmu CKY został przedstawiony na Rysunku 1.3. Sprawdzana gramatyka musi zostać sprowadzona do postaci normalnej Chomsky’ego (forma, w której wszystkie produkcje danej gramatyki zostały sprowadzone do postaci: (A→a) lub (B→CD), gdzie duże litery symbolizują symbole terminalne, natomiast małe litery symbolizują symbole nieterminalne).[9]

Każda gramatyka bezkontekstowa która w wyniku swoich reguł nie produkuje symbolu pustego może zostać sprowadzona do postaci normalnej Chomsky’ego.



Rysunek 1.5 - pseudokod probabilistycznego algorytmu CKY [10]

Jak widać na Rysunku 1.3, przy pojednczej analizie składniowej z wykorzystaniem parsera CKY występują 3, zagnieżdżone w sobie pętle przebiegające na długość sprawdzanego słowa, sprawia to, że zależność złożoności obliczeniowej tego algorytmu od długości sprawdzanego zdania jest równa n3 , co oznacza, że n-krotne zwiększenie długości sprawdzanego zdania spowoduje n3 krotne wydłużenie czasu potrzebnego na przeprowadzenie analizy składniowej.

# BADANE JĘZYKI

W eksperymentach wykorzystałem dwa przykładowe języki nieregularne przedstawione w artykule: „*GA-based Learning of Context-Free Grammars using Tabular Representations.*” [11], oraz jeden język obejmujący sekwencje aminokwasowe rodziny motywów białkowych.

## PIERWSZY JĘZYK TESTOWY

Pierwsza gramatyka opisuje język: nad alfabetem: {a, b, c}.

Oznacza to, że wszystkie słowa należące do tego języka składają się z symboli: „a”, „b” oraz „c”, ułożonych w następującej sekwencji:

* na początku każdego słowa znajduje się *n* powtórzeń symbolu „a”,
* po symbolach „a” znajduje się taka sama liczba symboli „b”,
* każde słowo zakończone jest dowolną, niezerową liczbą powtórzeń symbolu „c”.

Gramatyka opisująca ten język, w postaci normalnej Chomsky’ego, składa się z:

* 3 symboli terminalnych: {a, b, c},
* 6 symboli nieterminalnych: {S, T, V, A, B, C}, spośród których symbol „S” jest symbolem startowym,
* 8 reguł: {S → TC, S → SC, T → AV, T → AB, V → TB, A → a, B → b, C → c}.

Można zauważyć, że w przyjętej gramatyce, żaden z symboli nieterminalnych, zdolnych do generowania symbolu terminalnego nie jest zdolny do generowania innego symbolu nieterminalnego, natomiast żaden z symboli nieterminalnych zdolnych do generowania innego symbolu nieterminalnego nie jest zdolny do generowania jakiegokolwiek symbolu terminalnego.

Ze względu na to, symbole nieterminalne można podzielić na:

* Symbole nieterminalne leksykalne (symbole nieterminalne generujące symbole terminalne): {A, B, C},
* symbole nieterminalne strukturalne (symbole nieterminalne generujące symbole nieterminalne): {S, T, V}.

Analogiczne nazewnictwo stosuję w przypadku reguł, które zostały podzielone na reguły leksykalne (generujące symbole terminalne) oraz reguły strukturalne (generujące symbole nieterminalne).

Na potrzeby testowania maszynowego uczenia, dla powyższej gramatyki, wygenerowano gramatykę pokrywającą, czyli gramatykę zawierającą reguły umożliwiające generowanie dowolnego zestawu symboli nieterminalnych z każdego symbolu strukturalnego oraz każdego symbolu terminalnego z dowolnego symbolu leksykalnego.[]

Gramatyka pokrywająca została dodatkowo rozszerzona o jeden dodatkowy symbol strukturalny: „U” w związku z czym zawierała 9 reguł leksykalnych oraz 196 reguł strukturalnych.

Gramatyka ta będzie w dalszej części pracy nazywana gramatyką testową 1, natomiast opisana wcześniej gramatyka przez nią rozszerzana: gramatyką docelową 1.

Podczas wszystkich symulacji, wykonywanych na gramatyce testowej 1 lub gramatyce docelowej 1, jako zbiór uczący, wykorzystano minimalny, wystarczający do nauczenia pierwszej gramatyki docelowej, zbiór próbek pozytywnych (należących do języka opisywanego przez tę gramatykę). Zbiór ten został wygenerowany za pomocą narzędzia dostępnego na stronie: „http://lukasz.culer.staff.iiar.pwr.edu.pl/gencreator.php” i zawierał 7 sekwencji: {„abc”, „aabbcc”, „aaabbbc”, „abcc”, „abccc”, „aabbcc”, „aaaabbbbc”}.[12]

Przy pomocy tego samego narzędzia wygenerowałem zbiór 100 sekwencji należących do sprawdzanego języka, oraz 100 sekwencji do niego nienależących. Zbiory te zostały wykorzystane jako zbiór walidacyjny przy ocenie poprawności nauki.

Średnia długość sekwencji, w przypadku zbioru sekwencji pozytywnych wynosi 16.7, natomiast w przypadku sekwencji negatywnych 12,41. Liczba wystąpień symboli: „a”, „b” i „c” wynosi: 418:418:794 w przypadku zbioru sekwencji pozytywnych, oraz 412:408:421 w przypadku zbioru sekwencji negatywnych.

Oba zbiory znajdują się w załączniku jako: „positiveTestGrammar1.txt” oraz „negativeTestGrammar1.txt”.

## DRUGI JĘZYK TESTOWY

Druga gramatyka opisuje język: nad alfabetem: {a, b, c}.

Oznacza to, że wszystkie słowa należące do tego języka składają się z symboli: „a” i „c” lub „b” i „c”, ułożonych w następującej sekwencji:

* na początku każdego słowa znajduje się jedno powtórzenie symbolu „a” lub „b”,
* po pierwszym symbolu występuje dowolna, niezerowa liczba powtórzeń symbolu „c”.

Gramatyka opisująca ten język, w postaci normalnej Chomsky’ego, składa się z:

* 3 symboli terminalnych: {a, b, c},
* 3 symboli nieterminalnych: {S, T, C}, spośród których symbol „S” jest symbolem startowym,
* 5 reguł: {S → TC, S → SC, T → a, T → b, C → c}.

Podobnie jak w przypadku pierwszej gramatyki testowej i docelowej, symbole nieterminalne oraz reguły zostały podzielone na strukturalne oraz leksykalne.

Dla powyższej gramatyki, nazywanej od tej pory gramatyką docelową 2, wygenerowano 4 rozszerzające je gramatyki pokrywające:

* do pierwszej z nich nie dodano żadnych dodatkowych symboli nieterminalnych,
* do drugiej dodano dodatkowy nieterminalny symbol strukturalny: „U”,
* do trzeciej dodano dodatkowy nieterminalny symbol leksykalny: „A”,
* do czwartej dodano dodatkowy nieterminalny symbol strukturalny: „U”, oraz dodatkowy nieterminalny symbol leksykalny: „A”.

Gramatyki te będą od teraz nazywane kolejno: „gramatyką testową 2.1”, „gramatyką testową 2.2”, „gramatyką testową 2.3” oraz „gramatyką testową 2.4”.

Podobnie jak przy pierwszym języku testowym, za pomocą tego samego narzędzia, wygenerowano zbiory uczące i walidujące wykorzystywane później we wszystkich doświadczeniach przeprowadzanych dla gramatyk drugiego języka testowego.

Zbiór uczący składał się z pięciu sekwencji: {„ac”, „bc”, „acc”, „bcc”, „accc”}. Zbiór walidujący składał się z 10 sekwencji należących do języka i 100 sekwencji do niego nienależących.

Średnia długość sekwencji, w przypadku zbioru sekwencji pozytywnych wynosi 6.5, natomiast w przypadku sekwencji negatywnych 11,82. Liczba wystąpień symboli: „a”, „b” i „c” wynosi: 5:5:55 w przypadku zbioru sekwencji pozytywnych, oraz 400:381:401 w przypadku zbioru sekwencji negatywnych.

Oba zbiory dostępne w załączniku jako: „positiveTestGrammar2.txt” oraz „negativeTestGrammar2.txt”.

## JĘZYK OPISUJĄCY SEKWENCJE NALEŻĄCE DO RODZINY MOTYWÓW BIAŁKOWYCH

Badany w doświadczeniu motyw białkowy jest odpowiedzialny za wiązanie wapnia i manganu. Białka należące do badanej grupy pochodzą z lektyn roślin strączkowych. [13]

Próbka pozytywna składała się z 24 sekwencji zawierających ten motyw białkowy, zaś próbka negatywna składała się ze 100 sekwencji niezawierających tego motywu. Wszystkie sekwencje należące zarówno do próbki pozytywnej jak i negatywnej mają długość 27.

Ze względu na niewielką liczbę sekwencji które zawierają badany motyw białkowy, oraz to, że proces nauki gramatyki wymaga dużego i różnorodnego zbioru uczącego, próbka pozytywna została podzielona na 4 części zawierające po 6 sekwencji, następnie utworzono 4 zestawy zbiorów uczących i walidujących, w których jedna część próbki pozytywnej wraz z próbką negatywną utworzyła zbiór walidujący, natomiast pozostałe pozytywne sekwencje stanowiły zbiór uczący:

* w zestawie pierwszym do zbioru walidacyjnego trafiła pierwsza część zbioru pozytywnego,
* w zestawie drugim do zbioru walidacyjnego trafiła druga część zbioru pozytywnego,
* w zestawie trzecim do zbioru walidacyjnego trafiła trzecia część zbioru pozytywnego,
* w zestawie czwartym do zbioru walidacyjnego trafiła czwarta część zbioru pozytywnego.

Ze względu na to, że aminokwasy zwyczajowo oznaczane są wielkimi literami, w przypadku tej gramatyki symbole terminalne były przedstawiane przez wielkie litery alfabetu łacińskiego, natomiast symbole nieterminalne przez cyfry.

Na potrzeby maszynowego uczenia utworzyłem gramatykę pokrywającą która zawierała:

* 20 symboli terminalnych oznaczonych przez litery odpowiadające standardowym jednoliterowym oznaczeniom aminokwasów
* 7 symboli nieterminalnych oznaczonych jako cyfry od 0 do 6, spośród których symbol oznaczony jako 0 jest jednocześnie symbolem startowym.

Symbole nieterminalne od 0 do 3 zostały wykorzystane jako nieterminalne symbole strukturalne, natomiast symbole „4”, „5” oraz „6” zostały wykorzystane jako nieterminalne symbole leksykalne.

# ZASTOSOWANY ALGORYTM MASZYNOWEGO UCZENIA

Wykorzystany w maszynowym uczeniu algorytm został stworzony w środowisku MATLAB i jest oparty na opisanym wcześniej prostym algorytmie genetycznym. Wartości wyznaczane podczas procesu uczenia to prawdopodobieństwa poszczególnych reguł sprawdzanej gramatyki.

Program jako dane wejściowe przyjmuje:

* zestaw reguł sprawdzanej gramatyki,
* zbiór sekwencji które mają zostać użyte jako zbiór uczący,
* zbiór sekwencji pozytywnych (należących do języka) i negatywnych (nienależących do języka), które łącznie tworzą zbiór walidujący,
* listę należących do danej gramatyki symboli terminalnych wraz z częstotliwością ich występowania względem siebie (wykorzystywane przy wyznaczaniu prawdopodobieństwa wygenerowania danej sekwencji przez model zerowy podczas procesu walidacji)
* parametry funkcji uczenia maszynowego (nazywane dalej: „hiperparametrami”) do których należą: liczba osobników w populacji (n), prawdopodobieństwo mutacji (pm), maksymalna skala mutacji (sm), oraz prawdopodobieństwo krzyżowania (pk),
* czas zegarowy (w godzinach) po upływie którego program zapisuje dotychczasowe wyniki (można podać kilka wartości).

Każdy osobnik jest tutaj wektorem liczb z przedziału od 0 do 1, długość wektora odpowiada liczbie reguł w danej gramatyce. Podczas inicjowania populacji początkowej program losuje prawdopodobieństwa reguł dla wszystkich osobników jako losową liczbę z rozkładu równomiernego w przedziale od 0 do 1.

W przypadku probabilistycznych gramatyk bezkontekstowych konieczne jest zadbanie o to, aby suma prawdopodobieństw wszystkich reguł o tym samym poprzedniku wynosiła 1. Losowo przypisane prawdopodobieństwa reguł zazwyczaj nie spełniają tego warunku. Aby poradzić sobie z tym problemem, program po rozlosowaniu prawdopodobieństw odpowiednio je skaluje (poprzez dzielenie prawdopodobieństwa sprawdzanej reguły (P(r)) przez sumę prawdopodobieństw wszystkich reguł mających tego samego poprzednika).

Przystosowanie poszczególnych osobników jest wyznaczane jako logarytm naturalny ze średniego prawdopodobieństwa z jakim sekwencje ze zbioru uczącego należą do testowanego języka (ln(Pśr)). Przystosowanie najlepszego osobnika w danej populacji jest oznaczane jako: „max(ln(Pśr))”, natomiast maksymalna wartość przystosowania osiągana podczas całej symulacji jest oznaczana jako: „max(max(ln(Pśr)))” lub: „wartość maksymalna z max(ln(Pśr))”. Analiza składniowa w wyniku której wyznaczane są prawdopodobieństwa z jakimi każda sekwencja należy do sprawdzanego języka odbywa się według opisanego wcześniej algorytmu CKY. Ze względu na to, że analiza składniowa jest najbardziej czasochłonną częścią programu, oraz to, że przystosowania poszczególnych osobników są od siebie całkowicie niezależne, program umożliwia równoległe wyznaczanie funkcji przystosowania dla wielu osobników jednocześnie przy wykorzystaniu kilku rdzeni (wszystkie symulacje opisane w pracy zostały wykonane przy użyciu 20 rdzeni co umożliwiło równoległe wyznaczanie funkcji przystosowania dla 20 osobników).

Proces reprodukcji przebiega zgodnie z założeniami prostego algorytmu genetycznego, jednak tworzona w jego wyniku populacja (nazywana od tej pory: „populacjąT1”) jest o połowę mniejsza od populacji bazowej. Populacja T1 jest następnie poddawana mutacjom. Ze względu na to, że prawdopodobieństwa reguł nie są wartościami binarnymi (a zgodnie z założeniami prostego algorytmu genetycznego cechy osobników powinny być binarne), podczas procesu mutacji, dla każdej cechy dla której doszło do mutacji, losowana jest liczba z przedziału od 0 do maksymalnej skali mutacji (jeden z hiperparametrów programu). Następnie, z jednakowym prawdopodobieństwem, wylosowana liczba jest dodawana lub odejmowana od aktualnego prawdopodobieństwa danej reguły (jeżeli ma dojść do odjęcia wylosowanej liczby, która jest większa od aktualnego prawdopodobieństwa danej reguły to jest ona zerowana). W wyniku zerowania niektórych reguł może dojść do sytuacji, w której dany osobnik nie jest w stanie zakwalifikować któregoś słowa z próbki uczącej do sprawdzanego języka (prawdopodobieństwo z jakim to słowo należy do języka jest równe 0), mutacje takie zostają uznane za krytycznie niekorzystne a,. osobnik ten nie zostaje przeniesiony do kolejnej populacji (zostaje zastąpiony przez swoją kopię sprzed mutacji).

Po procesie reprodukcji i mutacji do powstałej w ten sposób populacji T1 dodawana jest lepiej przystosowana połowa poprzedniej populacji (w niezmienionej formie), powstała w ten sposób populacja stanowi populację bazową w kolejnym cyklu nauki.

Proces nauki zostaje zakończony po przejściu przez, zadaną jako parametr wejściowy algorytmu, liczbę cykli.

Podczas nauki program zapisuje w pamięci takie parametry jak: średnie prawdopodobieństwo z jakim sekwencje z próbki uczącej należą do języka opisywanego przez sprawdzaną gramatykę, różnorodność populacji (rozumiana jako średnia z odległości euklidesowych każdego osobnika względem wszystkich pozostałych osobników w populacji), czas jaki upłynął w poszczególnych cyklach nauczania, liczba osobników u których doszło do mutacji krytycznie niekorzystnych. Ponadto, w każdym cyklu zapisywany jest najlepiej przystosowany osobnik (tworzą one historię najlepiej przystosowanych osobników), natomiast co 100 cykli zapisywana jest cała populacja (zapisane w ten sposób populacje tworzą historię populacji).

Po zakończeniu procesu nauczania przeprowadzany jest proces oceny efektów nauczania (walidacji). W tym celu, dla co dziesiątej próbki z historii najlepszych osobników, wyznaczane są prawdopodobieństwa z jakimi sekwencje ze zbioru walidacyjnego (utworzonego z połącznia próbki pozytywnej i negatywnej które są parametrami wejściowymi programu) należą do sprawdzanego języka. Na podstawie uzyskanych w ten sposób prawdopodobieństw wykreślana jest krzywa ROC dla której wyznaczane jest ograniczone przez nią pole pod krzywą będące wskaźnikiem efektywności modelu predykcyjnego uzyskanego w wyniku uczenia.

# PRZEBIEG EKSPERYMENTU

## WPŁYW HIPERPARAMETRÓW ALGORYTMU MASZYNOWEGO UCZENIA NA PRZEBIEG I WYNIKI NAUKI

W pierwszej części eksperymentu zbadałem, w jaki sposób zmiana wartości hiperparametrów algorytmu uczącego wpływa na przebieg nauki i jej końcowe wyniki.

Podczas eksperymentu wykonałem szereg symulacji dla gramatyki testowej 1, z różnymi zestawami hiperparametrów, przedstawionymi w Tabeli 4.1.

Tabela 4.1 - wykorzystane zestawy hiperparametrów

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| l.p. | liczba osobników | prawdopodobieństwo mutacji | maksymalna skala mutacji | prawdopodobieństwo krzyżowania |
| 1 | 40 | 0,001 | 0,5 | 0,9 |
| 2 | 0,1 |
| 3 | 0,5 |
| 4 | 0,9 |
| 5 | 0,1 |
| 6 | 0,0001 | 0,5 |
| 7 | 0,01 |
| 8 | 80 | 0,001 | 0,5 | 0,9 |
| 9 | 0,1 |
| 10 | 0,5 |
| 11 | 0,9 |
| 12 | 0,1 |
| 13 | 0,0001 | 0,5 |
| 14 | 0,01 |
| 15 | 160 | 0,001 | 0,5 | 0,9 |
| 16 | 0,1 |
| 17 | 0,5 |
| 18 | 0,9 |
| 19 | 0,1 |
| 20 | 0,0001 | 0,5 |
| 21 | 0,01 |

Przed przeprowadzeniem symulacji dla pierwszej gramatyki testowej, w celu poznania wartości prawdopodobieństw poszczególnych reguł jak najbardziej zbliżonych do wartości optymalnych, przeprowadzono 3 symulację dla pierwszej gramatyki docelowej (hiperparametry ustawione jak w zestawie nr 3, liczba cykli po której zakończono naukę ustawiona na 1000).

Dla wszystkich przeprowadzonych symulacji, końcowa wartość średniego prawdopodobieństwa z jakim zdania ze zbioru uczącego należą do sprawdzanego języka (Pśr), jest praktycznie identyczna i po zlogarytmowaniu logarytmen naturalnym wynosi: ln(max(Pśr)) = -2.55.

Końcowe wartości prawdopodobieństw dla poszczególnych reguł we wszystkich testach również ostatecznie stają się do siebie bardzo podobne:

Tabela 4.2 - końcowe prawdopodobieństwa przypisane do poszczególnych reguł

|  |  |  |  |
| --- | --- | --- | --- |
| reguła | test1 | test2 | test3 |
| S→TC | 0,616 | 0,615 | 0,616 |
| S→SC | 0,384 | 0,385 | 0,384 |
| T→AV | 0,530 | 0,529 | 0,528 |
| T→AB | 0,470 | 0,471 | 0,472 |
| V→TB | 1,000 | 1,000 | 1,000 |
| A→a | 1,000 | 1,000 | 1,000 |
| B→b | 1,000 | 1,000 | 1,000 |
| C→c | 1,000 | 1,000 | 1,000 |

Dla wszystkich przeprowadzonych symulacji sprawdzano:

* jaka jest maksymalna wartość max(ln(Pśr)) w danej symulacji,
* po ilu cyklach max(ln(Pśr)) osiąga 95% maksymalnej wartości,
* po ilu minutach max(ln(Pśr)) osiąga 95% maksymalnej wartości,
* po jak wielu cyklach klasyfikator staje się bliski idealnemu (pole pod krzywą ROC większe lub równe 0,95),
* po ilu minutach klasyfikator staje się bliski idealnemu,
* po ilu cyklach każdy nieterminalny symbol leksykalny posiada tylko jedną regułę o niezerowym prawdopodobieństwie,
* jak wiele reguł, u najlepiej przystosowanego osobnika populacji, zostało wyzerowane w cyklu w którym max(ln(Pśr)) osiąga 95% maksymalnej wartości.

Wyniki dla wszystkich symulacji można znaleźć w Tabeli 1 w załączniku.

Symulacje dla których wartość max(ln(Pśr)) jest większa lub równa -2.8 uznałem za udane, ponieważ jest to wynik zbliżony do uzyskanego w symulacjach z wykorzystaniem gramatyki docelowej 1. Część symulacji osiągnęła jednak wartość max(ln(Pśr)) mniejszą niż -2.8, jest to spowodowane tym, że podczas procesu uczenia populacja utknęła w lokalnym maksimum które było znacznie poniżej maksimum globalnego, takie symulacje uznałem za nieudane.

W populacjach początkowych średnia wartość max(ln(Pśr)) wynosi około -9,5, najniższa końcowa wartość max(ln(Pśr)) uzsyskana w symulacji uznanej za nieudaną to -4,28 (uzyskana w symulacji nr 3 dla 6 zestawu hiperparametrów), zaś 18 z 23 nieudanych symulacji osiąga wartość max(ln(Pśr)) nie mniejszą niż -3.9, pokazuje to, że nawet pomimo utknięcia populacji w maksimum lokalnym, nauka spowodowała zwiększenie wartości max(ln(Pśr)) o kilka rzędów.

Warto również zwrócić uwagę, że w zdecydowanej większości przypadków (21 spośród 23), gramatyki uzyskane w wyniku symulacji uznanych za nieudane mogą wciąż pełnić rolę bardzo dobrych klasyfikatorów, świadczy o tym to, że w każdej symulacji po pewnej liczbie cykli pole pod krzywą ROC osiąga wartość co najmniej 0,95.

W Tabeli 2 znajdującej się w załączniku przedstawiono ile symulacji, dla każdego zestawu parametrów, zostało uznane za udane, oraz średnie wartości najważniejszych parametrów wraz z odchyleniem standardowym (przy liczeniu średniej i odchylenia standardowego brane pod uwagę były wyłącznie symulacje uznane za udane).

### Wpływ liczby osobników na przebieg i wyniki nauki

Każda sprawdzana liczba osobników w populacji była testowana przy wykorzystaniu 7 zestawów hiperparametrów (dla każdej liczby osobników zestawy testowe były takie same). Porównanie liczby symulacji uznanych za udane, maksymalnej wartości max(ln(Pśr)) oraz czasu po którym max(ln(Pśr)) osiągał 95% maksymalnej wartości, dla każdej sprawdzanej liczby osobników w populacji zostało przedstawiłem w Tabeli 4.3.

Tabela 4.3 - Porównanie wyników symulacji dla zestawów o różnej liczebności populacji

|  |  |  |  |
| --- | --- | --- | --- |
| **liczba osobników** | **40** | **80** | **160** |
| liczba symulacji | 21 | 21 | 21 |
| liczba udanych symulacji | 14 | 10 | 16 |
| odsetek udanych symulacji | 66,67% | 47,62% | 76,19% |
| max(max(ln(Pśr))) | -2,50 | -2,49 | -2,50 |
| t (0,95\*max(Pśr))[cykle] | 1397,64 | 1219,20 | 1262,81 |
| t (0,95\*max(Pśr))[min] | 10,80 | 15,13 | 26,68 |

Odsetek udanych symulacji jest najwyższy dla symulacji, w których populacja liczyła 160 osobników, a najniższy przy populacjach liczących 80 osobników. Pokazuje to, że, w przypadku gramatyki testowej 1, skłonność wykorzystywanego narzędzia do osiadania w maksimach lokalnych nie jest silnie powiązana z liczbą osobników w populacji. Ponadto widać, że populacja licząca 40 osobników jest wystarczająco liczna, aby uzyskać wyniki, które będą, z dużym prawdopodobieństwem, bardzo zbliżone do optymalnych.

Średnia wartość max(max(ln(Pśr))) z udanych symulacji jest bardzo zbliżona dla wszystkich sprawdzanych liczebności populacji, co pokazuje, że zwiększanie liczby osobników nie ma wyraźnego wpływu na końcowy efekt procesu nauczania dla gramatyki testowej 1.

Średnia liczba cykli nauki potrzebna, aby max(ln(Pśr)) osiągnęło 0,95 maksymalnej wartości, jest największa dla symulacji z populacjami liczącymi 40 osobników. Jednakże czas trwania każdego cyklu jest zależny od liczby osobników w populacji, w związku z czym, czas zegarowy, potrzebny, aby max(ln(Pśr)) osiągnęło 0,95 maksymalnej wartości, jest zauważalnie najmniejszy dla populacji liczącej 40 osobników.

### Wpływ prawdopodobieństwa mutacji na przebieg i wyniki nauki

W celu zbadania wpływu prawdopodobieństwa mutacji na przebieg i wyniki maszynowego uczenia wydzieliłem spośród wszystkich symulacji trzy grupy:

* grupa pierwsza – symulacje dla zestawów nr: 3, 6 oraz 7,
* grupa druga – symulacje dla zestawów nr: 10, 13 oraz 14,
* grupa trzecia – symulacje dla zestawów nr: 17, 20 oraz 21.

W grupie pierwszej znajdują się zestawy różniące się względem siebie tylko prawdopodobieństwem mutacji, zaś liczba osobników we wszystkich zestawach z tej grupy wynosi 40. Podobnie, w grupie drugiej i trzeciej, znajdują się zestawy o różnych prawdopodobieństwach mutacji, zaś liczba osobników w tych grupach to 80 oraz 160.

W Tabelach 4.3 – 4.6 zestawiłem uśrednione wartości sprawdzanych parametrów (analogicznie jak w punkcie 4.1.1) dla grup 1-3, oraz dla sumy zestawów o tym samym prawdopodobieństwie mutacji z grup 1-3 (z wyłączeniem czasu zegarowego i liczby cykli po których max(ln(Pśr)) osiąga 0,95 maksymalnej wartości, ponieważ parametr ten jest zależny od liczby osobników w populacji, w związku z czym należy go analizować w każdej grupie osobno).

Tabela 4.4 - porównanie wyników symulacji dla zestawów o różnym prawdopodobieństwie mutacji, grupa 1

|  |  |  |  |
| --- | --- | --- | --- |
| **prawdopodobieństwo mutacji** | **0,01** | **0,001** | **0,0001** |
| liczba symulacji | 3 | 3 | 3 |
| liczba udanych symulacji | 2 | 3 | 0 |
| odsetek udanych symulacji | 66,67% | 100,00% | 0,00% |
| max(max(ln(Pśr))) | -2,55 | -2,48 | - |
| t (0,95\*max(Pśr))[cykle] | 820 | 1256 | - |
| t (0,95\*max(Pśr))[min] | 6,15 | 9,92 | - |

Tabela 4.5 - porównanie wyników symulacji dla zestawów o różnym prawdopodobieństwie mutacji, grupa 2

|  |  |  |  |
| --- | --- | --- | --- |
| **prawdopodobieństwo mutacji** | **0,01** | **0,001** | **0,0001** |
| liczba symulacji | 3 | 3 | 3 |
| liczba udanych symulacji | 1 | 1 | 1 |
| odsetek udanych symulacji | 33,33% | 33,33% | 33,33% |
| max(max(ln(Pśr))) | -2,48 | -2,40 | -2,68 |
| t (0,95\*max(Pśr))[cykle] | 529 | 1458 | 2559 |
| t (0,95\*max(Pśr))[min] | 6,74 | 17,34 | 32,39 |

Tabela 4.6 - porównanie wyników symulacji dla zestawów o różnym prawdopodobieństwie mutacji, grupa 3

|  |  |  |  |
| --- | --- | --- | --- |
| **prawdopodobieństwo mutacji** | **0,01** | **0,001** | **0,0001** |
| liczba symulacji | 3 | 3 | 3 |
| liczba udanych symulacji | 3 | 2 | 3 |
| odsetek udanych symulacji | 100,00% | 66,67% | 100,00% |
| max(max(ln(Pśr))) | -2,50 | -2,50 | -2,50 |
| t (0,95\*max(Pśr))[cykle] | 620 | 791 | 2138,333 |
| t (0,95\*max(Pśr))[min] | 13,12 | 18,03 | 44,09 |

Tabela 4.7 - porównanie wyników symulacji dla zestawów o różnym prawdopodobieństwie mutacji, suma

|  |  |  |  |
| --- | --- | --- | --- |
| **prawdopodobieństwo mutacji** | **0,01** | **0,001** | **0,0001** |
| liczba symulacji | 9 | 9 | 9 |
| liczba udanych symulacji | 6 | 6 | 4 |
| odsetek udanych symulacji | 66,67% | 66,67% | 44,44% |
| max(max(ln(Pśr))) | -2,52 | -2,47 | -2,55 |

Odsetek udanych symulacji jest wyraźnie wyższy, gdy prawdopodobieństwo mutacji wynosiło 0,01 lub 0,001, wskazuje to, że zbyt niskie prawdopodobieństwo mutacji (0,0001) jest niekorzystne dla procesu uczenia i sprzyja osiadaniu w lokalnych maksimach. Populacja o wysokiej liczbie osobników (160) radzi sobie zauważalnie lepiej, od mniej licznych populacji, z bardzo wysokim (0,01) oraz bardzo niskim (0,001) prawdopodobieństwem mutacji. We wszystkich trzech grupach wyraźnie widać, że nauka przebiega tym szybciej, im prawdopodobieństwo mutacji jest większe. Średnie wyniki max(max(ln(Pśr))) są zbliżone dla wszystkich sprawdzanych prawdopodobieństw mutacji.

### Wpływ maksymalnej skali mutacji na przebieg i wyniki nauki

W celu zbadania wpływu maksymalnej skali mutacji na przebieg i wyniki maszynowego uczenia, analogicznie jak w punkcie 3.1.2, wydzieliłem spośród wszystkich symulacji trzy grupy:

* grupa pierwsza – symulacje dla zestawów nr: 3, 4 oraz 5,
* grupa druga – symulacje dla zestawów nr: 10, 11 oraz 12,
* grupa trzecia – symulacje dla zestawów nr: 17, 18 oraz 19.

Tabela 4.8 - porównanie wyników symulacji dla zestawów o różnych maksymalnych skalach mutacji, grupa 1

|  |  |  |  |
| --- | --- | --- | --- |
| **maksymalna skala mutacji** | **0,1** | **0,5** | **0,9** |
| liczba symulacji | 3 | 3 | 3 |
| liczba udanych symulacji | 2 | 3 | 1 |
| odsetek udanych symulacji | 66,67% | 100,00% | 33,33% |
| max(max(ln(Pśr))) | -2,46 | -2,48 | -2,55 |
| t (0,95\*max(Pśr))[cykle] | 2602 | 1256 | 767 |
| t (0,95\*max(Pśr))[min] | 19,48 | 9,92 | 6,71 |

Tabela 4.9 - porównanie wyników symulacji dla zestawów o różnych maksymalnych skalach mutacji, grupa 2

|  |  |  |  |
| --- | --- | --- | --- |
| **maksymalna skala mutacji** | **0,1** | **0,5** | **0,9** |
| liczba symulacji | 3 | 3 | 3 |
| liczba udanych symulacji | 0 | 1 | 2 |
| odsetek udanych symulacji | 0,00% | 33,33% | 66,67% |
| max(max(ln(Pśr))) | - | -2,40 | -2,48 |
| t (0,95\*max(Pśr))[cykle] | - | 1458 | 1131 |
| t (0,95\*max(Pśr))[min] | - | 17,34 | 13,86 |

Tabela 4.10 - porównanie wyników symulacji dla zestawów o różnych maksymalnych skalach mutacji, grupa 3

|  |  |  |  |
| --- | --- | --- | --- |
| **maksymalna skala mutacji** | **0,1** | **0,5** | **0,9** |
| liczba symulacji | 3 | 2 | 2 |
| liczba udanych symulacji | 3 | 2 | 2 |
| odsetek udanych symulacji | 100,00% | 100,00% | 100,00% |
| max(max(ln(Pśr))) | -2,53 | -2,50 | -2,44 |
| t (0,95\*max(Pśr))[cykle] | 1840,333 | 791 | 876 |
| t (0,95\*max(Pśr))[min] | 38,05 | 18,03 | 18,34 |

Tabela 4.11 - porównanie wyników symulacji dla zestawów o różnych maksymalnych skalach mutacji, suma

|  |  |  |  |
| --- | --- | --- | --- |
| **maksymalna skala mutacji** | **0,1** | **0,5** | **0,9** |
| liczba symulacji | 9 | 9 | 9 |
| liczba udanych symulacji | 5 | 6 | 5 |
| odsetek udanych symulacji | 55,56% | 66,67% | 55,56% |
| max(max(ln(Pśr))) | -2,50 | -2,47 | -2,48 |

Odsetek udanych symulacji i maksymalna wartość max(max(ln(Pśr))) są minimalnie wyższe w przypadku maksymalnej skali mutacji równej 0,5. W grupach 1 i 2 wyraźnie widać, że zwiększenie maksymalnej skali mutacji zmniejsza czas potrzebny, aby max(ln(Pśr)) osiągnęło 0,95 maksymalnej wartości, w przypadku grupy trzeciej widać to przy zmianie skali mutacji z 0,1 na 0,5, jednak przy zmianie z 0,5 do 0,9 czas nauki wydłuża się. Bardzo niska maksymalna skala mutacji (0,1) ma wyraźnie niepożądany wpływ na proces uczenia wydłużając go i pogorszając ostateczne wyniki, wyniki uzyskiwane dla maksymalnej skali mutacji 0,5 oraz 0,9 są do siebie zbliżone.

### Wpływ prawdopodobieństwo krzyżowania na przebieg i wyniki nauki

W celu zbadania wpływu maksymalnej skali mutacji na przebieg i wyniki maszynowego uczenia, analogicznie jak w punkcie 3.1.2, wydzieliłem spośród wszystkich symulacji trzy grupy:

* grupa pierwsza – symulacje dla zestawów nr: 1, 2 oraz 3,
* grupa druga – symulacje dla zestawów nr: 8, 9 oraz 10,
* grupa trzecia – symulacje dla zestawów nr: 15, 16 oraz 17.

Tabela 4.12 - porównanie wyników symulacji dla zestawów o różnym prawdopodobieństwie krzyżowania, grupa 1

|  |  |  |  |
| --- | --- | --- | --- |
| **prawdopodobieństwo krzyżowania** | **0,1** | **0,5** | **0,9** |
| liczba symulacji | 3 | 3 | 3 |
| liczba udanych symulacji | 3 | 3 | 3 |
| odsetek udanych symulacji | 100,00% | 100,00% | 100,00% |
| max(max(ln(Pśr))) | -2,48 | -2,48 | -2,49 |
| t (0,95\*max(Pśr))[cykle] | 1372,66 | 1256 | 1356,66 |
| t (0,95\*max(Pśr))[min] | 10,44 | 9,92 | 10,72 |

Tabela 4.13 - porównanie wyników symulacji dla zestawów o różnym prawdopodobieństwie krzyżowania, grupa 2

|  |  |  |  |
| --- | --- | --- | --- |
| **prawdopodobieństwo krzyżowania** | **0,1** | **0,5** | **0,9** |
| liczba symulacji | 3 | 3 | 3 |
| liczba udanych symulacji | 2 | 1 | 3 |
| odsetek udanych symulacji | 66,67% | 33,33% | 100,00% |
| max(max(ln(Pśr))) | -2,44 | -2,40 | -2,50 |
| t (0,95\*max(Pśr))[cykle] | 1260,5 | 1458 | 954,33 |
| t (0,95\*max(Pśr))[min] | 15,11 | 17,34 | 12,30 |

Tabela 4.14 - porównanie wyników symulacji dla zestawów o różnym prawdopodobieństwie krzyżowania, grupa 3

|  |  |  |  |
| --- | --- | --- | --- |
| **prawdopodobieństwo krzyżowania** | **0,1** | **0,5** | **0,9** |
| liczba symulacji | 3 | 2 | 2 |
| liczba udanych symulacji | 1 | 2 | 2 |
| odsetek udanych symulacji | 33,33% | 100,00% | 100,00% |
| max(max(ln(Pśr))) | -2,55 | -2,50 | -2,51 |
| t (0,95\*max(Pśr))[cykle] | 817 | 791 | 1129 |
| t (0,95\*max(Pśr))[min] | 19,06 | 18,03 | 24,69 |

Tabela 4.15 - porównanie wyników symulacji dla zestawów o różnym prawdopodobieństwie krzyżowania, suma

|  |  |  |  |
| --- | --- | --- | --- |
| **prawdopodobieństwo krzyżowania** | **0,1** | **0,5** | **0,9** |
| liczba symulacji | 9 | 9 | 9 |
| liczba udanych symulacji | 6 | 6 | 8 |
| odsetek udanych symulacji | 66,67% | 66,67% | 88,89% |
| max(max(ln(Pśr))) | -2,48 | -2,47 | -2,50 |

Serie z największym prawdopodobieństwem krzyżowania mają najwyższy odsetek udanych symulacji, jednak wartości max(ln(Pśr)) we wszystkich grupach są do siebie zbliżone, nie widać również żadnego widocznego powiązania pomiędzy prawdopodobieństwem krzyżowania a czasem potrzebnym by max(ln(Pśr)) osiągnęło 95% maksymalnej wartości.

## WPŁYW LICZBY SYMBOLI NIETERMINALNYCH NA WYNIKI NAUKI

W drugiej części doświadczenia zbadałem wpływ liczby nieterminalnych symboli strukturalnych oraz leksykalnych na złożoność gramatyki będącej wynikiem uczenia maszynowego.

Po przeanalizowaniu wyników pierwszej części eksperymentu, zdecydowałem się wykorzystać zestaw hiperparamterów, który został oznaczony w poprzedniej części doświadczenia jako zestaw nr 3.

Korzystając z tego zestawu hiperparametrów przeprowadziłem po 3 symulacje dla:

* gramatyki testowej 2.1 (G1),
* gramatyki testowej 2.2 (G2),
* gramatyki testowej 2.3 (G3),
* gramatyki testowej 2.4 (G4).

Dla każdej symulacji sprawdzałem jaka jest maksymalna uzyskana wartość max(ln(Pśr)), oraz dla jak wielu reguł po zakończeniu nauki jest przypisane prawdopodobieśtwo:

* równe 0%,
* mniejsze od 1%,
* mniejsze od 10%,
* większe od 10%,
* większe od 30%,
* równe 100%.

Wyniki tych symulacji zostały zestawione w Tabeli 4.16.

Tabela 4.16 - zestawienie wyników wszystkich symulacji dla drugiego języka testowego

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| gramatyka | nr symulacji | max(ln(Pśr)) | r(p=0%) | r(p<1%) | r(p<10%) | r(p>10%) | r(p>30%) | r(p=100%) |
| G1 | 1 | -3,73448 | 10 | 10 | 11 | 10 | 3 | 0 |
| 2 | -3,60708 | 12 | 12 | 14 | 7 | 4 | 0 |
| 3 | -3,60904 | 11 | 11 | 12 | 9 | 3 | 0 |
| G2 | 1 | -3,55327 | 40 | 40 | 42 | 9 | 6 | 1 |
| 2 | -3,55372 | 41 | 41 | 43 | 8 | 5 | 1 |
| 3 | -3,55761 | 35 | 36 | 43 | 8 | 5 | 1 |
| G3 | 1 | -1,78362 | 30 | 30 | 30 | 8 | 6 | 1 |
| 2 | -1,78419 | 30 | 30 | 30 | 8 | 6 | 1 |
| 3 | -1,78382 | 29 | 29 | 31 | 7 | 6 | 1 |
| G4 | 1 | -1,78358 | 56 | 61 | 72 | 9 | 7 | 1 |
| 2 | -1,90963 | 43 | 49 | 72 | 9 | 7 | 1 |
| 3 | -1,72794 | 72 | 72 | 73 | 8 | 8 | 2 |

Wszystkie 3 symulacje dla gramatyki G1 cechują się najniższą wartością max(ln(Pśr)), symulacje dla gramatyki G3 osiągają zauważalnie wyższe wartości max(ln(Pśr)) niż symulacje dla gramatyki G2, pokazuje to, że w badanym przypadku, rozszerzenie gramatyki o dodatkowy nieterminalny symbol strukturalny było o wiele bardziej korzystne niż rozszerzenie jej o dodatkowy symbol leksykalny.

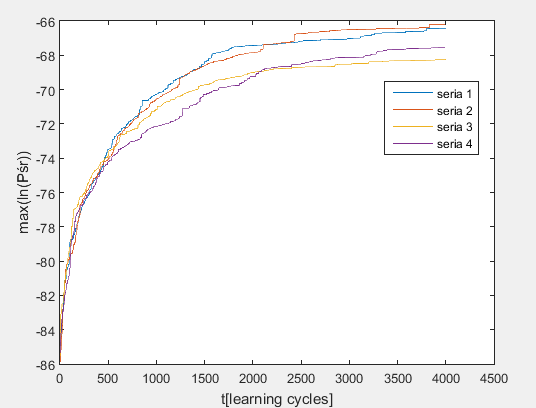
## TESTOWANIE MASZYNOWEGO UCZENIA NA JĘZYKU OBEJMUJĄCYM SEKWENCJE AMINOKWASOWE RODZINY MOTYWÓW BIAŁKOWYCH

W ostatniej części doświadczenia, polegającej na przetestowaniu stworzonego algorytmu na języku który opisuje rzeczywistą rodzinę białek, zdecydowałem się na ponowne użycie zestawu hiperparametrów wykorzystanych w doświadczeniu 4.2.

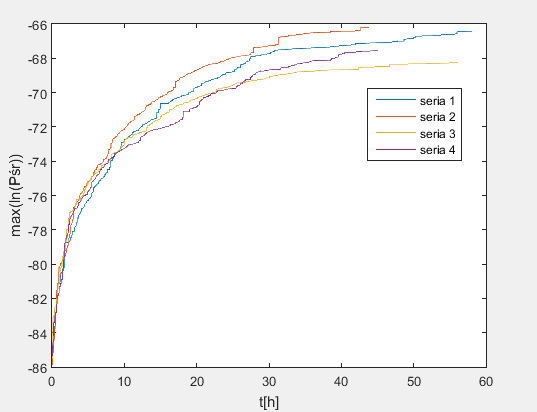
W doświadczeniu wykonane zostały 4 serie:

* seria 1 wykorzystuje pierwszy zestaw testowy i walidacyjny,
* seria 2 wykorzystuje drugi zestaw testowy i walidacyjny,
* seria 3 wykorzystuje trzeci zestaw testowy i walidacyjny,
* seria 4 wykorzystuje czwarty zestaw testowy i walidacyjny.

Na rysunku 4.1 widać jak zmieniała się wartość max(ln(Pśr)) w poszczególnych cyklach nauki dla wszystkich czterech serii.

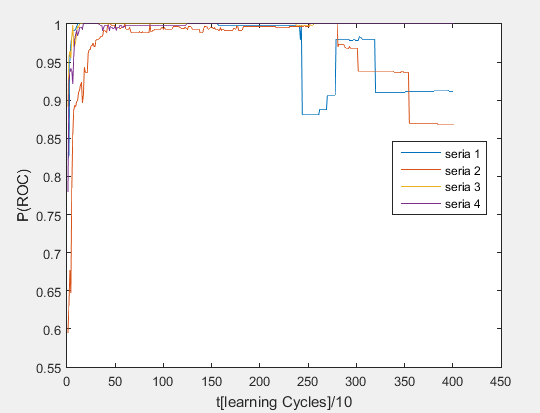


Rysunek 4.1 - wyniki maszynowego uczenia dla czterech serii doświadczeń na sekwencjach białkowych (w cyklach nauki)Na rysunku 4.2 przedstawiono jak wartość max(ln(Pśr)) zależy od zegarowego czasu nauki (w godzinach).



Rysunek 4.2 - wyniki maszynowego uczenia dla czterech serii doświadczeń na sekwencjach białkowych (w godzinach)

Jak widać podczas trwania nauki wartość max(ln(Pśr)) cały czas rośnie, jeśli spojrzymy jednak na to, jak zmieniało się pole pod krzywą ROC podczas nauki możemy zauważyć, że w przypadku serii pierwszej oraz drugiej wykres w kilku momentach gwałtownie opada (Rysunek 4.3).



Rysunek 4.3 – zmiana wartości pola pod krzywą ROC podczas maszynowego uczenia

Jest to wynik tak zwanego przeuczenia, gramatyka zaczyna dopasowywać się tak dokładnie do próbki uczącej, że część pozytywnych próbek ze zbioru walidującego zaczyna być klasyfikowana jako nienależąca, lub należąca z bardzo małym prawdopodobieństwem, do testowanej gramatyki [14].

Zmiany wartości max(ln(Pśr)) oraz pola pod krzywą ROC dla poszczególnych serii są przedstawione w Tabelach 4.16 oraz 4.17.

Tabela 4.17

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | seria 1 | | seria 2 | |
| t [h] | max(ln(Pśr)) | P(ROC) | max(ln(Pśr)) | P(ROC) |
| 0 | -85,03 | 0,82 | -85,87 | 0,60 |
| 6 | -75,43 | 1,00 | -74,51 | 1,00 |
| 12 | -72,14 | 1,00 | -71,25 | 1,00 |
| 18 | -70,19 | 1,00 | -69,09 | 0,99 |
| 24 | -68,92 | 1,00 | -68,11 | 0,99 |
| 30 | -67,73 | 1,00 | -67,27 | 1,00 |
| 36 | -67,39 | 1,00 | -66,52 | 0,97 |
| 42 | -67,14 | 0,88 | -66,38 | 0,87 |
| end | -66,42 | 0,91 | -66,20 | 0,87 |

Tabela 4.18

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | seria 3 | | seria 4 | |
| t [h] | max(ln(Pśr)) | P(ROC) | max(ln(Pśr)) | P(ROC) |
| 0 | -85,52 | 0,89 | -85,32 | 0,78 |
| 6 | -74,72 | 1,00 | -75,02 | 1,00 |
| 12 | -72,37 | 1,00 | -72,87 | 1,00 |
| 18 | -70,70 | 1,00 | -71,55 | 1,00 |
| 24 | -69,71 | 1,00 | -69,80 | 1,00 |
| 30 | -69,14 | 1,00 | -68,69 | 1,00 |
| 36 | -68,75 | 1,00 | -68,15 | 1,00 |
| 42 | -68,65 | 1,00 | -67,64 | 1,00 |
| end | -68,26 | 1,00 | -67,54 | 1,00 |

# PODSUMOWANIE

Przeprowadzone na gramatykach testowych eksperymenty pozwoliły na określenie w jaki sposób zmiany hiperparametrów algorytmu, oraz liczba symboli terminalnych oraz nieterminalnych użytych przy generowaniu gramatyki pokrywającej wpływa na proces uczenia maszynowego.

Doświadczenie wykonane dla gramatyki opisującej język obejmujący sekwencje białkowe pokazało, że z pomocą stworzonego narzędzia możliwe jest uzyskanie gramatyki dobrze opisującej sprawdzaną rodzinę białek, która może pełnić funkcję bardzo dobrego klasyfikatora.

# LITERATURA

[1] D.B. Searl: A primer in macromolecular linguistics, Biopolymers. vol. 99,2013, s. 203–217.

[2] P.G. Higgs, T.K. Attwood: Bioinformatyka i ewolucja molekularna. PWN, Warszawa2008.

[3] Ghaheri A, Shoar S, Naderan M, Hoseini SS. The Applications of Genetic Algorithms in Medicine. Oman Med J. 2015;30(6):406-16.

[4] J. Arabas: Wykłady z algorytmów ewolucyjnych. WNT, Warszawa 2004.

[5] A.Osyczka, S.Kundu: A New Method to Solve Generalized Multicriteria Optimization Problems Using the Simple Genetic Algorithm, Structural Optimization.vol. 10, 1995, s. 95.

[6] Regresja liniowa jednej zmiennej, https://knbit.edu.pl/pl/wiki/ai/kurs-machine-learning/02-regresja-liniowa-jednej-zmiennej/

[7] J. E. Hopcroft, R. Motwani, J.D. Ullman: Wprowadzenie do teorii automatów, języków i obliczeń. PWN Warszawa 2005

[8] Cocke, John; Schwartz, Jacob T. 1970; Programming languages and their compilers: Preliminary notes

[9] N. Chomsky. On certain formal properties of grammars. Information and Control, 2(2):137 – 167, 1959. ISSN 0019-9958.

[10] https://gawron.sdsu.edu/compling/course\_core/assignments/prob\_parsing\_assignment.htm

[11] Sakakibara, Yasubumi and Mitsuhiro Kondo. “GA-based Learning of Context-Free Grammars using Tabular Representations.” ICML (1999).

[12] Unold, O., Culer, Ł., Kaczmarek, A. (2018). Iterative method of generating artificial context-free grammars. The 14th International Conference on Grammatical Inference (ICGI 2018, Wrocław)

[13] Tetko, I.V.; Livingstone, D.J.; Luik, A.I. Neural network studies. 1. Comparison of Overfitting and Overtraining, J. Chem. Inf. Comput. Sci., 1995, 35, 826-833

[14] N. Sharon and H. Lis. Legume lectins–a large family of homologous proteins. The FASEB Journal, 4(14):3198–3208, 1990. PMID: 2227211.