Tekst ten jest bezpośrednią kontynuacją wstępu teoretycznego który wysyłałem Panu już wcześniej, przesłał mi Pan już do niego Pańskie komentarze jednak wciąż nie zdążyłem wprowadzić niezbędnych poprawek, dlatego przesyłam tę część oddzielnie.

1. Wykorzystane gramatyki

W eksperymentach zdecydowano się na wykorzystanie gramatyk opisujących dwa języki które zostały przedstawione w artykule: „*GA-based Learning of Context-Free Grammars using Tabular Representations.*” Yasubumiego Sakakibariego oraz Mitsuhirego Kondo.[], oraz jedną gramatykę opisującą białka.

* 1. Pierwsza gramatyka testowa

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”.

Utworzona na potrzeby eksperymentu gramatyka opisująca ten język, w postaci normalnej Chomsky’ego, składała 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}.

Jak widać, 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 zostały podzielone na:

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

Analogiczny podział zastosowano w przypadku reguł, które zostały podzielone na reguły leksykalne (generujące symbole terminalne) oraz reguły strukturalne (generujące symbole nieterminalne).

Na potrzeby 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 pierwszą gramatyką testową, natomiast opisana wcześniej gramatyka przez nią rozszerzana: pierwszą gramatyką docelową.

* 1. Druga gramatyka testowa

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”.

Utworzona na potrzeby eksperymentu gramatyka opisująca ten język, w postaci normalnej Chomsky’ego, składała 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 drugą gramatyką docelową, 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: „drugą gramatyką testową 1”, „drugą gramatyką testową 2”, „drugą gramatyką testową 3” oraz „drugą gramatyką testową 4”.

* 1. **Gramatyka opisująca białka**

No tutaj właśnie problem bo gdzieś mi się zapodział ten link do Pana artykułu, mógłbym prosić o ponowne podesłanie go?

1. **Zastosowany algorytm maszynowego uczenia**

Wykorzystany w maszynowym uczeniu algorytm jest oparty na opisanym wcześniej prostym algorytmie genetycznym.

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), zbiór powstający w wyniku ich połączeniem jest wykorzystywany jako zbiór walidujący,
* listę należących do danej gramatyki symboli terminalnych wraz z częstotliwością ich występowania (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).

Wartości wyznaczane podczas procesu uczenia to prawdopodobieństwa poszczególnych reguł sprawdzanej gramatyki, dlatego każdy osobnik jest tutaj wektorem liczb z przedziału od 0 do 1, długość każdego 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 probabilistycznych gramatykach bezkontekstowych konieczne jest zadbanie o to, aby suma prawdopodobieństw wszystkich reguł o tym samym poprzedniku wynosiła 1. Podczas losowania prawdopodobieństw poszczególnych reguł może dojść do niezachowania tego warunku. Aby poradzić sobie z tym problemem, program przechowuje w pamięci powiązania pomiędzy wszystkimi regułami i po rozlosowaniu prawdopodobieństw skaluje je w ten sposób, aby ich suma wynosiła 1 (poprzez dzielenie prawdopodobieństwa sprawdzanej reguły (P(r)) przez sumę prawdopodobieństw wszystkich reguł mających tego samego poprzednika):

Przystosowanie poszczególnych osobników jest wyznaczane jako średnie prawdopodobieństwo z jakim sekwencje ze zbioru uczącego należą do testowanego języka. Analiza składniowa w wyniku której wyznaczane są prawdopodobieństwa z jakimi każda sekwencja należy do sprawdzanego języka odbywa się według opisanego wcześniej algorytmu CKY.

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.

1. **Przebieg eksperymentu**
   1. **Wpływ wartości hiperparametrów algorytmu maszynowego uczenia na jego przebieg i wyniki**

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

Podczas eksperymentu wykonano szereg symulacji dla pierwszej gramatyki testowej, z różnymi zestawami hiperparametrów, wykorzystane zestawy zestawiono w poniższej tabeli:

Tabela 3.1- Wykorzystane zestawy hiperparametrów

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| l.p. | n | pm | ms | pk |
| 1 | 40 | 0,001 | 0,5 | 0,5 |
| 2 | 80 | 0,0001 | 0,5 | 0,5 |
| 3 | 80 | 0,001 | 0,5 | 0,5 |
| 4 | 80 | 0,01 | 0,5 | 0,5 |
| 5 | 80 | 0,1 | 0,5 | 0,5 |
| 6 | 80 | 0,001 | 0,1 | 0,5 |
| 7 | 80 | 0,001 | 0,9 | 0,5 |
| 9 | 80 | 0,001 | 0,5 | 0,1 |
| 10 | 80 | 0,001 | 0,5 | 0,9 |
| 11 | 160 | 0,0001 | 0,5 | 0,5 |
| 12 | 160 | 0,001 | 0,5 | 0,5 |
| 13 | 160 | 0,01 | 0,5 | 0,5 |
| 14 | 160 | 0,1 | 0,5 | 0,5 |
| 15 | 160 | 0,001 | 0,1 | 0,5 |
| 16 | 160 | 0,001 | 0,9 | 0,5 |
| 17 | 160 | 0,001 | 0,5 | 0,1 |
| 18 | 160 | 0,001 | 0,5 | 0,9 |
| 19 | 320 | 0,001 | 0,5 | 0,5 |

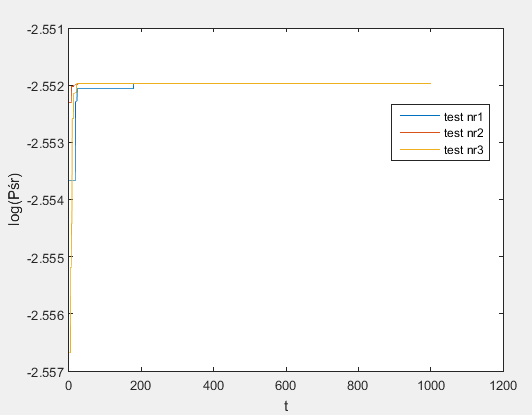
Podczas wszystkich symulacji, jako zbiór uczący, wykorzystano minimalny, wystarczający do nauczenia pierwszej gramatyki docelowej, zbiór próbek pozytywnych (należących do języka opisywanego przez tę gramatykę).

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”}.

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

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 12, 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 wynosi: log(Pśr) = -2.55.



Rysunek 1 - zmiana wartości log(Pśr) podczas nauki

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

|  |  |  |  |
| --- | --- | --- | --- |
| reguła | test1 | test2 | test3 |
| S→TC | 0,615505 | 0,61495 | 0,616091 |
| S→SC | 0,384495 | 0,38505 | 0,383909 |
| T→AV | 0,530415 | 0,529287 | 0,528155 |
| T→AB | 0,469585 | 0,470713 | 0,471845 |
| V→TB | 1 | 1 | 1 |
| A→a | 1 | 1 | 1 |
| B→b | 1 | 1 | 1 |
| C→c | 1 | 1 | 1 |

Dla wszystkich przeprowadzonych symulacji sprawdzano: jaka jest maksymalna wartość log(Pśr) w danej symulacji, po ilu cyklach log(Pśr) osiąga 95% maksymalnej wartości, po ilu sekundach log(Pśr) osiąga 95% maksymalnej wartości, po jak wielu cyklach klasyfikator staje się bliski idealnemu (pole pod krzywą ROC większe lub równe 0,95), po jak długim czasie klasyfikator staje się bliski idealnemu, jak wiele reguł u najlepiej przystosowanego osobnika ostatniej populacji ma prawdopodobieństwo: równe 0, mniejsze od 1%, mniejsze od 10%, większe od 10%, większe od 30%, równe 100%, wyniki zostały zestawione w poniższych tabelach:







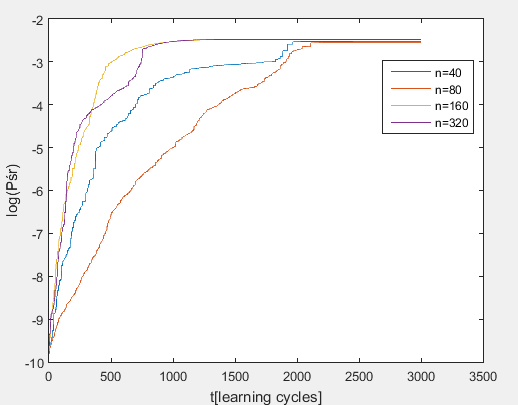




Zdecydowałem się na poszerzenie zakresu sprawdzanych zestawów parametrów, dlatego część kolumn pozostaje pusta, symulacje dla nich wciąż się wykonują.

Symulacje dla których wartość 0,95\*max(Pśr) jest większa od -2.8 można uznać za wystarczająco zbliżone do wyników uzyskanych dla symulacji z wykorzystaniem pierwszej gramatyki docelowej. Część spośród symulacji osiągnęła jednak wartość 0,95\*max(Pśr) mniejszą niż -2.8, jest to spowodowane tym, że podczas procesu uczenia populacja utknęła w lokalnym maksimum które było znacznie gorsze od maksimum globalnego. Warto jednak zwrócić uwagę na to, że nawet wytworzone w wyniku tych symulacji gramatyki mogą wciąż pełnić rolę bardzo dobrych klasyfikatorów, świadczy o tym wysoki odsetek wyzerowanych reguł oraz to, że w każdej symulacji po pewnej liczbie cykli pole pod krzywą ROC osiąga wartość zbliżoną lub równą 1.

Porównując wykresy zależności log(Pśr) od liczby cyklów nauki możemy zauważyć, że symulacje w których populacje liczyły więcej osobników osiągają najlepsze możliwe dostosowanie szybciej, niż symulacje w których populacje liczyły mniej osobników:



Należy jednak zwrócić uwagę na to, że czas trwania cyklu nauki nie jest stały, zależy on od liczby osobników w danej populacji oraz od liczby niewyzerowanych prawdopodobieństw reguł w obrębie genotypu każdego osobnika, w związku z czym, istotne jest także przeanalizowanie wykresu przedstawiającego zależność log(Pśr) od czasu trwania nauki:

