Podstawy Sztucznej Inteligencji
Zespół: BT.TM: Karbownik Patrycja, Grabowski Kamil, Aksamit Sebastian

Dokumentacja

Zadanie:

Stworzenie własnego algorytmu (ewolucyjnego lub genetycznego) do rekonstrukcji drzewa filogenetycznego.

Informacje wstępne:

Drzewo filogenetyczne - graf acykliczny prezentujący drogę rozwoju, pochodzenie i zmiany ewolucyjne grupy organizmów (zazwyczaj gatunków); graf przedstawiający ewolucyjne zależności pomiędzy sekwencjami lub gatunkami wszystkich organizmów.

Porównywanie gatunków na podstawie sekwencji kwasów nukleinowych jest uznawane za wiarygodny i precyzyjny wyznacznik stopnia pokrewieństwa. Pozwala też wyznaczyć czas powstania mutacji / specjacji (proces biologiczny w wyniku którego powstają nowe gatunki).

Macierz substytucji - macierz kar i nagród za odpowiednie dopasowanie nukleotydów.

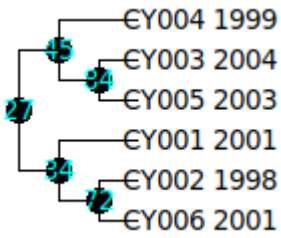
Opis problemu:

Ilość możliwych drzew zależy od ilości taksonów / liści (ich ilość oznaczymy jako *n)*. Istnieją drzewa ukorzenione, gdzie korzeń reprezentuje “praprzodka” (jest ich $\frac{(2n-3)!}{2^{n-2}*(n-2)!}$), a także drzewa nieukorzenione, których jest $\frac{(2n-5)!}{2^{n-3}*(n-3)!}$. Taka ilość możliwych drzew uniemożliwia sprawdzenie wszystkich możliwości w celu wybrania najlepszego rozwiązania już przy liczbie kilkunastu taksonów. Problem stworzenia takiego drzewa jest NP-zupełny.

W ramach pierwszego projektu zostanie zbadana skuteczność wykorzystywania algorytmu ewolucyjnego do tworzenia zadowalającego drzewa końcowego.

Założenia implementacji:

- Sposób przechowywania danych:
  - sekwencje genetyczne - pliki *.txt*
  - podobieństwo sekwencji - symetryczna macierz o rozmiarze *n x n*, gdzie *n* - ilość sekwencji oraz vector dostępności poszczególnych sekwencji (dostępność mówi o tym czy dana sekwencja została już dodana do drzewa)
  - macierz substytucji - symetryczna macierz o rozmiarze 5 x 5
  - drzewo - sekwencje określające konkretne organizmy będą przechowywane w węzłach zwanych *liściami* z atrybutami: *nazwa*, *rok*, opcjonalnie: *sekwencja*. Będą one łączone przy użyciu węzłów - *node*, które będą miały odniesienie do lewego i prawego poddrzewa.
- Sposób prezentacji drzewa - drzewo z liściami po prawej stronie ułożonymi w kolumnę, które są połączone wierzchołkami z dołączonymi do tychże wartościami ich prawdopodobieństwa wystąpienia (obliczanymi w czasie oceny drzewa), wizualizacja:



- Język programowania: Python
- Użyte biblioteki: *tkinter* (interfejs graficzny, biblioteka wbudowana Pythona) i *multiprocessing* dla efektywnego zrównoleglenia działania programu, *numpy*

Sposób uruchomienia programu:

Aby uruchomić program należy przez interpreter Pythona dla wersji 3 uruchomić *reconstruction.py*. Wymagane do poprawnego działania programu jest posiadanie bibliotek tkinter, numpy i multiprocessing.

Po uruchomieniu program wymaga podania liczby procesów - *k* równolegle uruchomionych algorytmów.

Po zakończeniu działania wyświetlone zostaje okno z narysowanymi drzewami. Klawiszami *a* i *d* można przełączać się między najlepszymi wynikami każdego z równoległych poszukiwań.

Dekompozycja:

Projekt podzielony jest na dziewięć plików:

- reconstruction.py - plik z głównym wywołaniem programu: ładowanie danych do pamięci i uruchomienie odpowiednich funkcji
- nodes.py - plik z klasami wierzchołków drzewa filogenetycznego
- symmetric_matrix.py - klasa z zaimplementowaną macierzą symetryczną przechowywaną jako tablica jednowymiarowa
- substitution_matrix.py - klasa macierzy substytucji, wykorzystująca macierz symetryczną, oraz z funkcjami przeznaczonymi do jej edycji w trakcie działania algorytmu ewolucyjnego
- phylogenetic_tree.py - plik z fukcjami tworzącymi drzewo filogenetyczne
- parallel.py - plik z funkcajami używanymi przy ocenie podobieństwa sekwencji
- alignment.py - plik z funkcjami używanymi do uliniawiania sekwencji
- scoring.py - plik z funkcjami wykorzystywanymi przy ocenie prawdopodobieństwa wystąpienia drzewa
- graphic_tree.py - plik z funkcjami potrzebnymi do stworzenia graficznego wyświetlenia drzewa

Algorytmy:

Główny algorytm:

1. Tworzymy *k* losowych macierzy substytucji i dla każdej z nich uruchamiamy algorytm ewolucyjny:
    a. *X* - wygenerowany pierwszy osobnik (pierwsze drzewo), *x* - odpowiadająca mu macierz substytucji, która jest modyfikowana przez algorytm ewolucyjny, wg algorytmu *"Tworzenie Drzewa Filogenetycznego"*
    b. Generujemy potomka *Y = X + sigma * N(0,1)*:
        i. wartości w komórkach macierzy zmieniane są przy użyciu parametru *sigma* oraz losowanych wartości z rozkładu normalnego *N(0,1)* - dla każdej komórki sigma jest stała, zaś różne są wartości wylosowane z rozkładu
    c. Oceniamy stworzone drzewo filogenetyczne (opis algorytmu w punkcie *"Ocena Drzewa"*) i wybieramy *X = f(X) > f(y) ? X : Y*, gdzie *f* to funkcja oceniająca
    d. Aktualizujemy parametr *fi* jako stosunek wybranych drzew *Y* w ciągu ostatnich *m* iteracji do liczby *m*
    e. Co *m*-ty krok zmień wartość sigma (*m* = 10):

$$sigma = \begin{cases} sigma * c1, jeśli\ fi < 1/5 \\ sigma * c2, jeśli\ fi > 1/5 \\ sigma, jeśli\ fi = 1/5 \end{cases}$$

    gdzie *c1* = 0,82, *c2* = 1,2
    f. Jeśli *sigma* < *sigma min*, zakończ algorytm zwracając *X* jako najlepsze drzewo, w przeciwnym razie wróć do punktu b

Tworzenie Drzewa Filogenetycznego:

1. Na podstawie stworzonej macierzy substytucji, uliniawiamy sekwencje genetyczne (opis algorytmu w punkcie *"Multiple Alignment - Heurystyczne uliniowienie wielu sekwencji z wykorzystaniem uliniawiwania par"*)
2. Tworzymy drzewo filogenetyczne:
    a. Obliczamy podobieństwo każdej pary sekwencji - uruchamiamy na nich uproszczony algorytm Needlemana-Wunscha obliczający ocenę uliniowienia (gdyby było one wykonywane), w którym przydzielamy nagrody lub kary za dopasowanie nukleotydów, oraz zapisujemy je w macierzy podobieństwa.
    b. Sprawdzamy ile zostało sekwencji do złączenia:
        i. Jeśli została jedna - kończymy przechodząc do pkt 4.
        ii. W przeciwnym razie - łączymy ze sobą dwie najbardziej podobne sekwencje - wprowadzamy korekcje w macierzy podobieństwa - podobieństwo połączonych sekwencji z innymi sekwencjami jest średnią arytmetyczną podobieństw między każdą z połączonych sekwencji a innymi, redukujemy macierz i wracamy do punktu 2b

Multiple Alignment:

Heurystyczne uliniowienie wielu sekwencji z wykorzystaniem uliniawiania par:

1. Wybierz sekwencję referencyjną:
   a. Dla każdej sekwencji oblicz sumę podobieństw między nią a innymi sekwencjami.
   b. Wybierz tą, której suma jest największa
2. Dopasuj wszystkie sekwencje z sekwencją referencyjną:
   a. dla każdej sekwencji wykonaj uliniowienie z sekwencją referencyjną przy pomocy algorytmu Needlemana-Wunscha i dodaj parę (*aligned_guide*, *aligned_sequence*) do listy *aligned_sequences*.
3. Złącz wszystkie uliniowione sekwencje:
   a. Weź pierwszą parę sekwencji z listy i dołącz *aligned_sequence* do "multiple alignment":
      i. Porównaj *aligned_guide* z nowo utworzonym *guide'm* (*multiple_guide*): Dla każdej kolumny:
         1. jeśli w żadnym z *guide'ów* nie występuje przerwa lub w obu występuje - w żadnej sekwencji nic nie zmieniamy.
         2. jeśli przerwa występuje w *multiple_guide* - dodaj do uliniawianej sekwencji przerwę.
         3. jeśli przerwa występuje w *aligned_guide* - do wcześniej uliniowionych sekwencji w danej kolumnie dodajemy przerwę.
      ii. Dodaj uliniawianą sekwencję do listy *multiple_aligned_sequences*.
   b. Jeśli złączono wszystkie sekwencje, zakończ działanie algorytmu i zwróć *multiple_aligned_sequences*. W przeciwnym przypadku wróć do punktu 3a.

Ocena Drzewa:

Opis ogólny:

Drzewo będzie oceniane przy użyciu metody znanej ze statystyki - bootstrapingu. Założenie jest takie, aby z wejściowych sekwencji, poprzez przemieszanie nukleotydów w konkretny sposób, otrzymać wiele sekwencji pochodnych, a następnie wygenerować z nich drzewa i zbadać w jakim procencie zależności z drzewa oryginalnego zostały zachowane w pochodnych.

Generowania nowych sekwencji

Są dane cztery sekwencje z indeksami przypisanymi do ich kolumn:

0	1	2	3	4	5	6
A	T	G	T	C	G	T
A	C	G	T	C	G	C
A	-	G	T	C	C	T
T	T	G	T	C	C	T

Zostają wygenerowane trzy nowe sekwencje, które wyglądają na przykład tak:

0	3	2	1	3	4	5
A	T	G	T	T	C	G
A	T	G	C	T	C	G
A	T	G	-	T	C	C
T	T	G	T	T	C	C

0	1	2	3	3	3	0
A	T	G	T	T	T	A
A	C	G	T	T	T	A
A	-	G	T	T	T	A
T	T	G	T	T	T	T

4	5	6	1	2	3	3
C	G	T	T	G	T	T
C	G	C	C	G	T	T
C	C	T	-	G	T	T
C	C	T	T	G	T	T

Warto zwrócić uwagę, że niektóre kolumny się powtarzają a niektórych nie ma - jest to efekt zamierzony, bowiem uwzględnia sytuację, gdzie odczyty mogły nie zawierać niektórych nukleotydów lub mogły je zduplikować. W procesie replikacji DNA przy tworzeniu nowego osobnika również może dojść do insercji, delecji czy inwersji.

Opis algorytmu

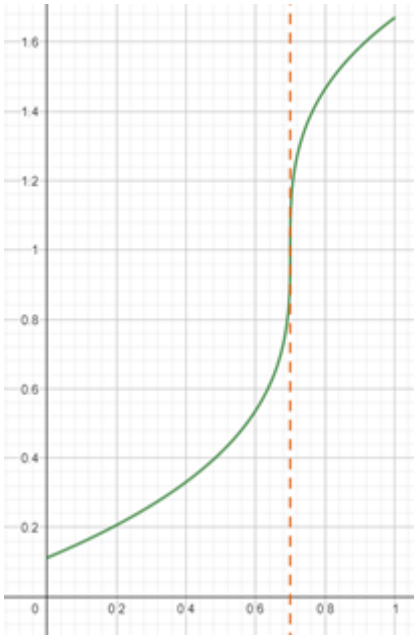
Jest dane drzewo skonstruowane przy użyciu pewnych danych wejściowych i macierzy substytucji. Nazywać je będziemy *X*. Sam algorytm będzie składał się z następujących kroków:

1. Z początkowych sekwencji utwórz *N* nowych grup sekwencji poprzez losowanie ze zwracaniem całych kolumn sekwencji (pokazane na przykładzie wyżej)
2. Dla każdej z *N* grup sekwencji skonstruuj nowe drzewo i umieść w grupie *Y*
3. Wybierz nieodwiedzony wierzchołek drzewa *X*
4. Zlicz ile razy wybrany wierzchołek występuje w drzewach *Y* a następnie podziel to przez *N* otrzymując w ten sposób **prawdopodobieństwo** wierzchołka
5. Jeżeli pozostały jeszcze nieodwiedzone wierzchołki w drzewie *X* wróć do punktu 3
6. Dla każdego wierzchołka wyznacz jego ocenę obliczając wartość funkcji f(x) dla *x* równego obliczonemu wcześniej **prawdopodobieństwu**
7. Zsumuj ocenę wszystkich wierzchołków i zwróć jako ocenę drzewa

Funkcja f(x) oceniająca wierzchołki

Formuła funkcji to $f(x) = \sqrt[3]{x - 0.7} + 1$

Kształt jest wynikową potrzeby faworyzowania wierzchołków o prawdopodobieństwie 0.7 lub większym. Wierzchołki o takim prawdopodobieństwie dla drzew filogenetycznych oznaczają wysokie prawdopodobieństwo prawdziwości takiego wierzchołka. Widać to dobrze na wykresie funkcji:



(Pomarańczowy pasek to x = 0.7)

Macierz podobieństwa:

Macierz podobieństwa będzie określać jak bardzo każda para sekwencji jest do siebie podobna - im wyższa ocena, tym większe podobieństwo.
Przykładowa ocena podobieństwa sekwencji:

Przykładowa macierz substytucji:

	A	G	C	T	-
A	10	-1	-3	-4	d
G	-1	7	-5	-3	d
C	-3	-5	9	0	d
T	-4	-3	0	8	d
-	d	d	d	d	2

d = -5
gdzie *d* - kara za przerwę

Sekwencje:

0	1	2	3	4	5	6	7	8
A	A	C	T	G	G	C	T	-
A	-	-	G	-	G	C	A	-

Ocena = 10 - 5 - 5 - 3 - 5 + 7 + 9 - 4 + 2 = 6

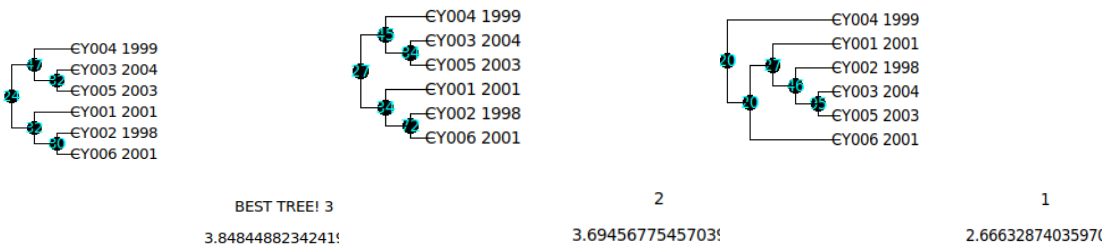
Sprawdzenie Identyczności Drzew:

Aby wykorzystać algorytm Bootstraping'u potrzeba sposobu porównywania drzew. W tym problemie zostało to rozwiązane tak, że konkretny liść *i* ma przypisaną wartość 2^i, natomiast wartość węzła jest równa sumie wartości swoich dzieci. Sprawdzenie identyczności drzew sprowadza się do tego, że sprawdzamy wartości węzłów. Jeśli wartości są różne, to drzewa na pewno nie są identyczne. Jeśli wartości są równe, to patrzymy czy ich dzieci też mają równe wartości. Jeśli nie, to drzewa są różne. Jeśli tak, to wywołujemy ten algorytm dla poszczególnych par poddrzew. Powtarzamy aż dojdziemy do liści lub na którymś etapie zostanie stwierdzone, że drzewa nie są identyczne.

Opis Wyników:

Zgodnie z oczekiwaniami w czasie działania programu - kiedy ten komunikuje na bieżąco aktualne wartości funkcji oceny - dostrzec można, że wyniki w istocie są coraz lepsze. Można również dostrzec słuszność założenia potrzeby użycia zrównoleglonego algorytmu 1+1 choćby na poniższym przykładzie:

Program został uruchomiony dla skromnej 6 elementowej paczki sekwencji o długości 13 po uliniowieniu. W ramach uruchomienia programu dla trzech równoległych procesów otrzymano trzy następujące wyniki:



Na trzy próby jedna okazała się mieć strukturę zdecydowanie różną od pozostałych dwóch. Osiągnęła również słabszy wynik, zatem jest to drzewo mniej prawdopodobne. Widać zatem zasadność zastosowania zrównoleglonej wersji algorytmu 1+1.

Plan Optymalizacji i Dalsze Usprawnienia:

W celu optymalizacji algorytmu można by np. spróbować zaimplementować inną wersję algorytmu Needlemana-Wunscha, w której przechowuje się w danej chwili tylko dwie kolumny z wynikami uliniawiania dwóch sekwencji - jest to bardzo duża optymalizacja pamięciowa.

Ważnym usprawnieniem mogłoby być również regularne tworzenie kopii zapasowych wyników w trakcie wykonywania się programu. Zrekonstruowanie drzewa posiadającego 6 sekwencji o długości (po uliniowieniu) 13 znaków zajęło programowi około 14 minut a nie jest to rozmiar nawet zbliżony do prawdziwych rozmiarów sekwencji (oczywiście maszyna testowa nie jest również wyspecjalizowana w wykonywaniu tego typu zadań). Jasnym jest więc, że dla poważnych zastosowań oprócz większej mocy obliczeniowej potrzebny byłby mechanizm regularnie zabezpieczający efekty pracy przed awarią.

Wartą rozważenia pod kątem usprawnień i optymalizacji byłaby też funkcja oceny. Po 1 jest to obecnie (po uproszczeniu innych algorytmów bio-informatycznych) najbardziej złożona obliczeniowo operacja całego procesu, ponieważ polega na wygenerowaniu wielu nowych drzew. Można by na przykład rozważyć inny sposób manipulacji danymi. Innym

jednak aspektem jest sama formuła f(x) pozwalająca ocenić wartość poszczególnych wierzchołków a w efekcie całego drzewa. Dostarcza ona wyniki zbliżone do pożądanych, jednak prawdopodobnie możliwe jest uzyskanie lepszych wyników dla lepszej formuły, być może bardziej faworyzującej wartości powyżej x = 0.7.