Nr 1 Marzec 2015

BioInfoWorld

Sekwencje Biologiczne

Koło Naukowe Bioinformatyki

W tym numerze:

BLAST jako podstawowy Bioalgorytm

Sekwencjonowanie DNA nowej generacji

DNA i RNA: Interpretacja i kompilacja

Symulowanie modelu matematycznego apokalipsy Zombie

Sztuczna inteligencja w Pythonie

BioInfo World

Nr 1 Marzec

Czasopismo Koła Naukowego Bioinformatyki BIT

ISSN

Redakcja

Rafał Szkotak
Paulina Kania
Rafał Klimek
Kamil Malisz
Marta Jaworska
Paweł Kaleciński
Karolina Cibor

Opiekun naukowy

dr Jacek Śmietański

Okładka

Kamil Malisz

Korekta

Marta Szkop

Skład

Rafał Szkotak

Wydawca

Koło Bioinformatyki BIT UJ

Kontakt

bioinfoworld.redakcja@gmail.com

Sfinansowane przez

Dziekan Wydziału Matematyki i Informatyki
Uniwersytetu Jagiellońskiego

Spis treści:

Tytułem wstępu

dr Jacek Śmietański 4

DNA i RNA: Interpretacja i kompilacja

Rafał Szkotak 7

Sztuczna inteligencja w Pythonie

Paulina Kania 10

Sekwencjonowanie nowej generacji - NGS

Rafał Szkotak 12

BLAST, jako podstawowy bioalgorytm

Paweł Kaleciński 13

Symulacja modelu matematycznego apokalipsy zombie

Rafał Klimek 15

Symulacje stochastyczne, czyli w poszukiwaniu różnorodności

Jan Majta 19

Sprawozdanie z działań Koła

Paulina Kania, Karolina Cibor 22

Słowniczek 24

Tytułem wstępu

Jacek Śmietański
Wydział Matematyki i Informatyki UJ
jacek.smietanski@ii.uj.edu.pl

Jeszcze nie tak dawno uważano, że w DNA człowieka zapisane jest ok. 100 tysięcy genów, że jeden gen koduje jedno białko i że tylko geny są użytecznymi biologicznie fragmentami genomu. Z czasem liczba szacowanych genów się zmniejszała: 30 tysięcy, 25, 22. Niedawno pojawiła się nawet praca sugerująca, że genów kodujących białka człowieka jest w genomie co najwyżej 19 tysięcy. W międzyczasie odkryto zjawisko alternatywnego splicingu, a po kilkunastu latach badań, ze zdumieniem stwierdzono, że nie jest to zjawisko wyjątkowe, ale proces, który dotyczyć może zdecydowanej większości genów kodujących białka. Okazało się również, że fragmenty DNA nie kodujące żadnych białek, dla których mocno utrwaliła się już nazwa „śmieciowe DNA", wcale takie śmieciowe nie są i ich rola nie ogranicza się tylko do zmniejszenia ryzyka mutacji w obszarach kodujących białko. Przede wszystkim bowiem, w rejonach tzw. „niekodujących" zawarta jest informacja o różnych cząsteczkach RNA, które – jak się okazuje – pełnią kluczową rolę w regulacji ekspresji genów i bez ich udziału musielibyśmy zapomnieć o produkcji białek i prawidłowym funkcjonowaniu komórki.

To stan wiedzy na dzień dzisiejszy. Jak widać, w ciągu ostatnich 20 – 30 lat, diametralnie zmieniły się nasze poglądy na temat ekspresji genów. I choć podstawowy schemat przepływu informacji w komórce, znany jako „centralny dogmat biologii molekularnej", który w pewnym uproszczeniu można przedstawić jako ciąg zdarzeń: „DNA --> mRNA --> białko" (informacja zawarta w DNA transkrybowana jest na mRNA, które ulega translacji do białka), wciąż pozostaje do pewnego stopnia aktualny, widzimy, że w rzeczywistości jest to schemat o wiele bardziej skomplikowany i nie taki liniowy, jak to się wcześniej wydawało.

Obserwujemy niewiarygodny wręcz rozwój technik sekwencjonowania. Jeszcze 35 lat temu uważano, że zsekwencjonowanie całego genomu człowieka to wyzwanie nierealistyczne, czas na to potrzebny szacowano na – bagatela – 4 miliony lat. Tymczasem dziś mówi się już o sekwencjonowaniu genomów w czasie zaledwie kilku godzin, przy koszcie nie przekraczającym kilkuset dolarów. Otwierają się przed na-

-mi niesamowite możliwości poznania nie tylko modelowych (czy też raczej przykładowych) genomów najróżniejszych gatunków organizmów, ale wręcz genomów całych populacji. To ogromna ilość informacji, której – czego tłumaczyć nie trzeba – bez udziału narzędzi i algorytmów bioinformatycznych, nie bylibyśmy w stanie w żaden sposób ogarnąć. Już samo przechowywanie naszej wiedzy w bazach danych i przeszukiwanie tych baz, stanowi nie lada wyzwanie. A to dopiero początek pracy. Jednym z głównych dalekosiężnych celów badań bioinformatycznych jest identyfikacja i ustalenie funkcji każdego białka w komórce oraz określenie oddziaływań i zależności pomiędzy różnymi białkami. Nie wszystko da się zrobić metodami eksperymentalnymi (a nawet, jeśli coś się da, to często koszty takich badań liczone są w milionach, a nawet miliardach dolarów). Może zatem techniki obliczeniowe pomogą? Wciąż wierzymy, że tak. Nieprzypadkowo zatem bioinformatyka ma bardzo silne powiązania z gałęziami nauki zwanymi „eksploracją danych", „sztuczną inteligencją", „uczeniem maszynowym". Wręcz ośmielę się wysnuć stwierdzenie, że wykorzystanie osiągnięć tych nauk jest podstawą i esencją badań bioinformatycznych.

Oddajemy w Wasze ręce pierwszy numer pisma BioInfoWorld, w którym chcemy przybliżyć istotę i ogromne znaczenie bioinformatyki we współczesnej nauce. Wiele artykułów będzie miało wymiar na wskroś praktyczny, opisując konkretne narzędzia czy biblioteki programistyczne, użyteczne w naszej codziennej pracy. Życzę Wam nie tylko przyjemnej lektury pisma, ale przede wszystkim inspiracji i efektywnego wykorzystania zawartej tu wiedzy w realizacji Waszych własnych projektów.

Koło Naukowe Bioinformatyki

Cichy Kącik Biojęzyków

Dział ten poświęcony jest rozmaitym zastosowaniom i implementacjom języków programowania w szeroko pojętej biologii. Można będzie tu znaleźć przeróżne informacje o bibliotekach Pythona, Ruby`ego, Perla, Javy i innych, których celem jest ułatwienie codziennego życia tzw. przyrodnikom. Najwięcej uwagi zostanie poświęconej Biopythonowi, a jako dodatek umieszczone zostaną informacje o tym, w jaki sposób na problem patrzą inne języki programowania.

DNA i RNA: Interpretacja i kompilacja

Rafał Szkotak

W pierwszym numerze czasopisma zamieszczone zostaną informacje o tworzeniu i obróbce **sekwencji** biologicznych, do których zaliczamy między innymi DNA i RNA, będące kombinacjami nukleotydów oraz peptydy i białka złożone z aminokwasów. Innymi słowy sekwencje są stringami, na które zostały nałożone ograniczenia związane z **alfabetami**. Podstawowe abecadło DNA składa się z czterech liter: A, C, T i G, będącymi jednoliterowymi skrótami zasad azotowych: adeniny, cytozyny, tyminy i guaniny, zatwierdzonymi przez IUPAC. RNA jest podobne do DNA, w którym zamiast T występuje U (uracyl). Białka i peptydy składają się z 20 aminokwasów tzw. kanonicznych, które są zakodowane bezpośrednio w RNA. Aminokwasy również posiadają swoje jednoliterowe skróty (patrz słowniczek: aminokwasy, nukleotydy). W Biopythonie [1] wszystkie te alfabety określane są mianem ***Unambiguous***. Piękno przyrody polega jednak na tym, że nie da się jej ująć w tak prostych ramach i od czasu do czasu na jednej pozycji w sekwencjach mogą zamiennie występować dwa lub więcej różnych nukleotydów (patrz słowniczek: mutacja punktowa). Podobnie jest z sekwencjami białek, gdzie na danej pozycji mogą wystąpić na przykład zamiennie glutamina i kwas glutaminowy. Można również nie wiedzieć, co właściwie znajduje się na danej pozycji. Biopython obsługuje takie przypadki dzięki alfabetom ***Ambiguous***. Dodatkowo na różnych pozycjach w łańcuchach mogą pojawić się aminokwasy, czy nukleotydy spoza kanonu. Wtedy należy użyć alfabetów ***Extended***.

Skoro sprawa alfabetu została wyjaśniona, można przejść do tworzenia obiektu *Seq* (***Bio.Seq.Seq***) Aby go stworzyć, potrzebujemy stringu z sekwencją oraz alfabetu (lista dostępnych alfabetów: ***Bio.Alphabet.IUPAC***, w którym zablokowane jest użycie małych liter oraz ***Bio.Alphabet***). Jeśli zabraknie drugiego argumentu, sekwencji *Seq* zostanie przypisany alfabet domyślny. Jak już wspomniano sekwencje działają jak standardowe pythonowe stringi, z których można pobierać wycinki, zwracać ich długość, łączyć (oczywiście nie ma możliwości połączenia białka i DNA), dzielić i w standardowy sposób rzutować pomiędzy typami. Sekwencje Seq, niestety, nie posiadają metody join. Problematyczne może być również porównywanie sekwencji, ponieważ aby proces miał sens użytkownik musi być pewnym, że są one stworzone z tych samych alfabetów oraz jawnie rzutować je na obiekty typu string. Zwykłe porównanie dwóch obiektów *Seq* (a==b) porówna ich adresy (które z konieczności są różne). Czym jeszcze różnią się obiekty *Seq* od stringów? Dzięki wbudowanym metodom pozwalają między innymi na stworzenie sekwencji komplementarnej do zawartej w obiekcie *Seq* (DNA, RNA), przeprowadzenie transkrypcji (DNA) oraz translacji lub odwrotnej transkrypcji (RNA), również przy użyciu niestandardowych kodów genetycznych (patrz słownik). Obiekty Seq mają jeszcze jedną ważną funkcjonalność. Są one nieedytowalne, podobnie jak znane z Pythona krotki, innymi słowy nie da się zasymulować na nich mutacji punktowych, delecji i insercji. Aby móc uzyskać możliwość substytucji nukleotydu lub aminokwasu należy rzutować obiekt typu Seq do ***MutableSeq*** lub po prostu stworzyć instancję obiektu *MutableSeq* w podobny sposób do tworzenia obiektu klasy *Seq*.

SeqRecord jest wyższą klasą w stosunku do *Seq*. Mówi się, że *Seq* dziedziczy od SeqRecord. Obiekt takiej klasy składa się z obiektu Seq zwierającego sekwencję z alfabetem (*.seq*), identyfikator sekwencji (*.id*), jej nazwę (*.name*), opis (*.description*), literową anotację (*.letter_annotation*), na przykład strukturę drugorzędową, anotację (*.annotation*) całej sekwencji oraz różne właściwości (*.features*). Minimalnym wymaganiem, które należy spełnić w celu stworzenia obiektu *SeqRecord* jest posiadanie obiektu klasy *Seq*, jako argumentu dla konstruktora. Innym sposobem tworzenia obiektu *SeqRecord* jest import sekwencji bezpośrednio z pliku fasta lub innych, do czego służy kolejny moduł *Bio.SeqIO*. Poszczególne pola obiektu (nazwa, opis) zostaną wypełnione automatycznie, niestety z powodu nieujednolicenia nagłówków plików w formacie fasta będzie to uzupełnienie niedosko-

nałe (wszystkie znaki od „>” do pierwszej spacji zostaną wpisane do pól .name oraz .id, cały nagłówek zaś do .description). Inne formaty zapisu sekwencji (takie jak GenBank) nie posiadają tego typu ograniczeń, o czym będzie mowa w następnym numerze, w którym szczegółowo omówimy moduł Seq.IO. Kolejny moduł, który dziedziczy po *SeqRecord*, niejako brat *Seq*, to ***SeqFeature***. Obiekty tego typu gromadzą opis sekwencji. Szczególnie biologom *features* kojarzyć się mogą z formatem GenBank. Pole to gromadzi informacje o źródle, pozycjach fragmentów kodujących białka, początku i końcu genu, czy sekwencji aminokwasowej odpowiadającej sekwencjom kodującym. Obiekty klasy SeqFeature stworzone są właśnie po to, aby zebrać jak najwięcej cech danego fragmentu obiektu *SeqRecord*. Do atrybutów obiektów *SeqFeatures* należą między innymi *.type* (na przykład cds). *SeqFeature* zawiera też informację o pozycji (*position*), która może być rozmyta (na przykład 100 > x i x > 35) i lokalizacji, która zawsze jest dokładna (od 1 do 35). Te zagadnienia zostały szczegółowo omówione w przewodniku po Biopythonie. Do ciekawych właściwości obiektów SeqRecord należy możliwość ich zapisu w różnych formatach, do czego służy wspomniany wcześniej moduł *Bio.SeqIO*. Informacje zawarte w strukturze obiektu zostaną automatycznie przeniesione do nagłówka pliku fasta lub do odpowiednich pól w formacie GenBank. Inną zaletą obiektów typu *SeqRecord* jest możliwość wycinania z nich interesującej nas części sekwencji, co więcej do tej nowopowstałej, skróconej sekwencji dopasowuje się obiekt *SeqFeatures* z nią stowarzyszony. Na obiektach typu *SeqRecord* możliwe jest również wykonywanie operacji *reverse complement.*

Jak wspomniano wyżej, druga część artykułu poświęcona zostanie pakietom BioJavy [2, 3] o podobnej funkcjonalności. Odpowiednikami modułów *Bio.Seq, Bio.Alphabet* oraz *Bio.SeqRecord* Pythona są pakiety ***org.biojava.bio.seq***, ***org.biojava.bio.symbo***l oraz ***org.biojava.bio.seqdb***. Sekwencje Javy zostały zaimplementowane w zupełnie inny sposób niż w Pythonie. Zamiast stringu sekwencja jest listą obiektów implementujących interfejs ***Symbol***. Tak więc każdy symbol w sekwencji posiada swoją nazwę, opis w postaci słownika oraz alfabet, z którego pochodzi. Ostatni atrybut instancji *Symbol* jest ważny dla niejednoznacznych liter, ponieważ A może oznaczać adeninę z DNA lub aminokwas alaninę. Podstawowymi operacjami zwracającymi kolejne atrybuty są: *getName()*, *getAnnotation()* oraz g*etMatches()*. Obiekty z interfejsem *Symbol* skupione są w listach implementują-

-cych interfejs *SymbolList*. Z każdym takim obiektem związany jest alfabet (naturalnie wszystkie symbole muszą być elementami tego samego alfabetu, zgodnego z alfabetem listy). Metody tego interfejsu pozwalają między innymi na zwrócenie: alfabetu (*getAlphabet()*), długości (*length()*), sekwencji jako stringa (*seqString()*), wycinka listy (*sublist(początek, koniec)*), wycinka stringa (*subString(początek, koniec)*), czy symbolu z danej pozycji (*SymbolAt(pozycja)*). Tutaj należy nadmienić, że numeracja symboli w liście rozpoczyna się od 1, a kończy na pozycji równej długości sekwencji, co jest zapewne naturalne dla biologów, ale niecodzienne dla informatyków. W przeciwieństwie do Pythona, BioJava na razie nie zaimplementowała mutacji sekwencji, tak więc niemożliwe jest zastąpienie jednego elementu listy innym. Oczywiście na obiektach *SymbolList* możliwa jest symulacja procesów biologicznych takich jak transkrypcja. BioJava pozwala również na użycie niejednoznacznych alfabetów nukleotydów lub aminokwasów, zatwierdzonych przez IUPAC. W celu wyświetlenia listy dostępnych alfabetów należy użyć metody *getAlphabets()*. Sama sekwencja również jest interfejsem. Z racji tego, że Sequeunce dziedziczy po SymbolList ma ona dostęp do wszystkich funkcjonalności tego drugiego. Dodatkowo obiekt *Sequence* ma możliwość przechowywania różnorodnych anotacji, zarówno całej sekwencji, jak i poszczególnych pozycji. Jest to możliwe dzięki implementacji kolejnych interfejsów: *Annotation*, *Features* oraz *StrandedFeatures*. Te ostatnie służą do opisu nici o znanym kierunku.

SŁOWNIK

DNA, RNA, aminokwasy kanoniczne i niekanoniczne, nukleotydy, transkrypcja, translacja, nić komplementarna, kod genetyczny, odwrotna transkrypcja, moduł, string, rzutowanie, mutacja, delecja, insercja, mut.punktowa, dziedziczenie, konstruktor, format fasta, format GenBank, gen, sekwencja kodująca(CDS), phred_quality, lista, instancja, interfejs

Bibliografia

[1] Chang J. i in., Biopython Tutorial and Cookbook, rozdziały 2 i 3, [dostęp 10.01.2015] oraz dokumentacja API odpowiednich modułów, [dostęp 10.01.2015]
[2] Prlic A. i in., “BioJava: an open-source framework for bioinformatics in 2012”, Bioinformatics 2012, [dostęp 10.02.2015]
[3] *http://biojava.org/wiki/BioJava:Tutorial*, dostęp 10.02.2015

Ciekawe funkcje Pythona

W niniejszym dziale zaprezntujemy Wam, drodzy Czytelnicy ciekawe biblioteki języka Python, na które natknęliśmy się w czasie studiów, a które mogą Wam się przydać.

Sztuczna inteligencja w Pythonie

Paulina Kania

Python jest językiem programowania wysokiego poziomu, który jest ciągle rozwijany jako projekt Open Source. Jest językiem dynamicznym i często jest też wykorzystywany jako język skryptowy. Jego zastosowania są bardzo różne: analiza danych, przetwarzanie tekstu, pisanie gier i różnych aplikacji, ale również ma szerokie zastosowanie w sztucznej inteligencji.

W poniższym artykule omówię niektóre z bibliotek Pythona, wspomagające implementację programów związanych ze sztuczną inteligencją. Można je podzielić na cztery główne kategorie:

- Biblioteki ogólnie związane ze sztuczną inteligencją
- Nauczanie maszynowe
- Język naturalny i przetwarzanie tekstu
- Sieci neuronowe

Spośród nich wybrałam te, które są najczęściej używane.

SimpleAI

Biblioteka *SimpleAI* zawiera większość algorytmów opisanych w książce *Artificial Intelligence, a Modern Approach* [1]. Opiera się ona częściowo na implementacji innej biblioteki AIMA, która zawiera wszystkie algorytmy z wyżej wymienionej publikacji. *SimpleAI*, jest pod tym względem bardziej uboga, ale z kolei jej autorom zależało na utworzeniu stabilnej i łatwej w utrzymaniu wersji.

Na chwilę obecną zawiera ona np. następujące algorytmy:

- Klasyfikacja statystyczna: algorytm związany z nauczaniem maszynowym. Może być użyty np. do rozpoznawania języka (system może nauczyć się rozpoznawać, czy piszemy do niego po angielsku, czy po polsku na podstawie statystycznej analizy występowania konkretnych liter).

• Algorytmy wyszukiwania: najpierw trzeba zdefiniować konkretny problem, a potem wybrać któryś z dostępnych algorytmów (DFS, BFS, algorytm zachłanny).
• Lokalne algorytmy wyszukiwania: ich idea jest taka sama jak zwykłych algorytmów wyszukiwania, ale różnią się wymaganiami implementacyjnymi.

PyBrain

PyBrain jest biblioteką, która oferuje elastyczne i łatwe w użyciu algorytmy służące do rozwiązywania problemów związanych z nauczaniem maszynowym. Rozwinięcie *PyBrain – Python-Based Reinforcement Learning, Artificial Intelligence and Neural Network Library* – oznacza, że jest to biblioteka oparta na Pythonie, dla uczenia przez wzmacnianie sztucznej inteligencji oraz sieci neuronowych. W porównaniu do innych bibliotek tego rodzaju, *PyBrain* cechuje się łatwością w użyciu nawet dla mniej zaawansowanych użytkowników, ale cechującą się równocześnie najnowocześniejszymi algorytmami wyszukiwania. Zawiera m.in. algorytmy do sieci neuronowych, uczenia przez wzmacnianie (oraz kombinacji tych dwóch), uczenia nadzorowanego i nienadzorowanego oraz do ewolucji. Kernel (jądro) *PyBraina* jest zbudowany w oparciu o sieci neuronowe, a wszystkie metody trenowania przyjmują na wejście sieć neuronową. To sprawia, że *PyBrain* jest dobrym narzędziem do skutecznego rozwiązywania problemów z życia wziętych. Dodatkowo *PyBrain* posiada narzędzia do tworzenia i wyświetlania wykresów, zapisywania i czytania formatu XML czy obsługi formatu netCDF (format, w którym dane oraz metadane zawarte są w tym samym zbiorze).

Scikit-learn

Scikit-learn jest biblioteką, stosowaną głównie do wyszukiwania i analizy danych. Jej algorytmy można podzielić na następujące kategorie:
• Klasyfikacja: służy do identyfikacji kategorii, do której należy obserwowane zjawisko. Zawiera algorytmy jak np. SVM, najbliższego sąsiada, losowych lasów drzew (random forest) stosowana w problemach typu detekcja spamu czy rozpoznawanie obrazów.
• Regresja: służy do przewidywania kolejnej wartości, np. w analizie problemu reakcji na leki lub cen akcji.
• Klastrowanie: automatyczne grupowanie podobnych do siebie elementów, np. w segmentacji rynku.

- Zmniejszenie wymiaru: polega na zmniejszeniu ilości zmiennych losowych branych pod uwagę, np. do wizualizacji.
- Selekcja modelu: porównanie, walidacja i wybór odpowiednich parametrów oraz modelu w zależności od problemu.
- Przetwarzanie: polega na ekstrakcji i normalizacji danych, np. żeby przekształcić zwykły tekst na dane do użycia w algorytmach uczenia maszynowego.

Feature Forge

Feature Forge zawiera narzędzia przydatne w zastosowaniach sztucznej inteligencji, w szczególności gdy korzysta się z biblioteki *scikit-learn*. *Feature Forge* ułatwia przede wszystkim definiowanie cech oraz ich testowanie w losowo generowanych przypadkach (tzw. *stress testing*). Pozwala to na stworzenie aplikacji bardziej odpornych na nieprawidłowe dane wejściowe. Ocena cech w danym zestawie danych opiera się na specjalnie utworzonej macierzy ewaluacji cech, którą można tak ustawić, żeby tolerowała też w pewnym stopniu niepoprawne dane. Ponadto bardzo ważną część tej biblioteki stanowi obszar do testowania: każde uruchomienie powoduje rejestrowanie, klasyfikację oraz odwzorowanie wcześniejszych eksperymentów w celu ustalenia najlepszych ustawień dla każdego nowego problemu.

Podsumowując, Python oferuje szeroką gamę darmowych bibliotek, przez co jest atrakcyjny w projektowaniu aplikacji wykorzystujących sztuczną inteligencję. Python zyskał na popularności dzięki dużej czytelności kodu oraz cechom języka skryptowego, które przydają się przy analizie dużych ilości danych przechowywanych głównie w plikach.

Bibliografia

[1] Russell S. J., Norvig P., *Artificial Intelligence, a Modern Approach*
[2] *https://wiki.python.org/moin/PythonForArtificialIntelligence*

Doniesienia z Frontu

Artykuły w tej rubryce opowiadać będą o najnowszych osiągnięciach i wyzwaniach bioinformatycznych XXI wieku.

Sekwencjonowanie nowej generacji - NGS

Rafał Szkotak

Opracowanie metod pozwalających na sekwencjonowanie DNA można uznać za pokonanie kolejnego kamienia milowego, informującego o coraz bliższym celu podróży. Naukowcy tacy jak Alan Maxam, Walter Gilbert (metoda Maxama i Gilberta 1977), a w szczególności Fryderyk Sanger (metoda Sangera 1977) na zawsze zapiszą się
w pamięci biologów różnych specjalizacji. Sekwencjonowanie nowej generacji wydaje się być kolejnym znakiem zbliżania się do istoty organizmów żywych, poznania ich genomów oraz transkryptomów.

Techniki nowej generacji takie jak sekwencjonowanie 454, czy Illumina umożliwiły przeprowadzenie niniejszych badań poprzez znaczne zwiększenie wydajności oraz niezwykle istotną redukcję kosztów. Do ważnych osiągnięć naukowych uzyskanych poprzez sekwencjonowanie nowej generacji zalicza się między innymi: możliwość poznania sekwencji genomowej mamuta [1, 2], a także neandertalczyka [3]. Sekwencjonowanie nowej generacji pozwala na szybkie i tanie odczytywanie sekwencji DNA na ogromną skalę.

W metodzie Illumina HiSeq 2000 możliwe jest uzyskiwanie w pojedynczym odczycie 200Gpz [4]. Wiąże się to z gromadzeniem olbrzymich ilości danych, których rozmiary nierzadko przekraczają terabajty (własna analiza), z których trzeba uzyskać biologicznie sensowne informacje. Postęp ten nie dotyczy jedynie sekwencji DNA, czy RNA, ale również białek. Coraz doskonalsze spektrometry masowe umożliwiają dokładniejsze (i tańsze) wnikanie w proteom komórki, co umożliwia między innymi porównywanie składu i ilości białek w różnych ich typach, czy stanach chorobowych. Tematyka proteomiczna rozwinięta zostanie w kolejnym numerze czasopisma.

Bibliografia

[1] Poinar HN, Schwarz C, Qi J, Shapiro B, Macphee RD, Buigues B, Tikhonov A, Huson DH, Tomsho LP, Auch A, Rampp M, Miller W, Schuster SC. *Metagenomics to paleogenomics: large-scale sequencing of mammoth DNA*. SCIENCE 2006; 311: 392-394

[2] Schuster SC. *Next-generation sequencing transforms today`s biology*. NAT METHODS 2007; 5(1): 16-18

[3] Green RE, Krause J i inni. *A draft sequence of neandertal genome*. SCIENCE 2010;
328:710-722

[4] *http://res.illumina.com, dostęp*: 3.06.2014

SŁOWNIK

Gpz

Bioalgorytmy i struktury danych

Ta część czasopisma poświęcona zostanie ciekawym algorytmom stosowanym powszechnie na całym świecie

BLAST, jako podstawowy bioalgorytm

Paweł Kaleciński

Czym jest algorytm?

Algorytm - pojęcie pozornie znane przez każdego, kto interesuję się w jakimś stopniu informatyką czy matematyką, lecz tak naprawdę ktoś zapytany o jego definicję raczej nie będzie w stanie jej podać. Według Wikipedii algorytm to skończony ciąg jasno zdefiniowanych czynności, koniecznych do wykonania pewnego rodzaju zadań.

Bioalgorytmy

A czym są bioalgorytmy? To zwykłe algorytmy wykorzystywane i zaimplementowane w sposób umożliwiający rozwiązywanie problemów bioinformatycznych oraz tworzenie odpowiednich narzędzi do pracy bioinformatyka.

Przejdźmy do rzeczy…

Artykuł otwierający cykl Bioalgorytmów i struktur danych poświecę na jeden z najpopularniejszych algorytmów wykorzystywanych w bioinformatyce, a mianowicie BLAST (ang.*Basic Local Alignment Search Tool*). Pierwszy raz został opublikowany w 1990 roku w *Journal of Molecular Biology*. Jego twórcami byli Stephen Altschul, Warren Gish, Webb Miller, Eugene Myers, i David J. Lipman z *National Institutes of Health*. Poprzednikiem BLASTa był powstały w 1985 algorytm FASTA.

BLAST jest podstawowym narzędziem każdego biologa. Służy do lokalnego przyrównywania sekwencji aminokwasów lub nukleotydów. Dzięki temu naukowcy są w stanie porównać zadaną sekwencje z innymi, zawartymi w bioinformatycznych bazach danych oraz ocenić ich podobieństwo. Serwery i bazy danych z jakich korzysta BLAST należą do NCBI (*The National Center for Biotechnology Information*).

Wyróżniamy różne rodzaje algorytmu:

- **PSI-BLAST** wyszukiwanie powtarzających się podobieństw sekwencji białka przy użyciu poszczególnych pozycji macierzy. Wyszukuje odległe ewolucyjnie, ale podobne sekwencje. Tworzy listę wszystkich ściśle związanych ze sobą białek. Białka te zbierane są w ogólną sekwencje „profile" , gdzie podsumowywane są istotne elementy obecne w sekwencjach.
- **RPS-BLAST** wyszukiwanie domen białka w Conserved Domains Database. Celem jest porównanie sekwencji do listy zachowanych motywów. Dzięki indeksowaniu baz danych, wzrosła prędkość wyszukiwania motywów od 10 do 100 razy w porównaniu do programu IMPALA.
- **BLASTN** mapowanie oligonukleotydów, powtórzeń, identyfikacja pokrewnych transkryptów. Zwraca najbardziej podobne sekwencje DNA z bazy danych DNA, którą określa użytkownik.
- **BLASTP** identyfikacja wspólnych rejonów między białkami, zbieranie białek. Wyszukuje sekwencje białek. Zwraca najbardziej podobne sekwencje białkowe z bazy danych białka, którą określa użytkownik.
- **BLASTX** szukanie kodujących sekwencji w genomach. Porównuje każdą z sześciu ramek odczytu produktów translacji badanej sekwencji nukleotydowej (obu nici) z bazą danych sekwencji protein.
- **TBLASTN** identyfikowanie transkryptów potencjalnie kodujących pokrewne białka (białka jeszcze nie w GenBanku), mapowanie białek do genomu. Program porównuje zapytanie białek z wszystkich sześciu ramkek odczytu bazy danych sekwencji nukleotydów
- **TBLASTX** przewidywanie genów na podstawie ortologów (z innego gatunku), poszukiwanie genów „gubionych" przez tradycyjne metody. Jest najwolniejszym z programów z rodziny BLAST. Tłumaczy sekwencje nukleotydów w każdej możliwej z sześciu ramek odczytu, w celu porównania ich z produktami translacji z sześciu ramek odczytu bazy danych sekwencji nukleotydów. TBLASTX może posłużyć do znalezienia bardzo odległych związków między sekwencjami nukleotydowymi.

Jak to działa?

Algorytm korzysta z metod heurystycznych, czyli opiera się na uproszczeniach, które maja na celu szybsze przeprowadzenie wyszukiwania. Porównuje tylko krótsze fragmenty sekwencji. Gdy znajdzie pierwsze dopasowanie globalne następuje przyrównanie sekwencji lo-

-kalnie. BLAST wykonuje przeszukanie wzorca próbując znaleźć ciągi takich samych liter i układa z nich najdłuższe słowa podobne do tych występujących w sekwencji kwerendy. Następnie skanowana jest baza danych sekwencji w celu znalezienia wzorca. Kiedy już mamy nasze porównanie używamy go do zainicjowania wydłużeń pozbawionych lub posiadających luki „słów". Gdy algorytm przeszuka wszystkie możliwe wzorce zawarte w sekwencji kwerendy i maksymalnie je rozszerzy, tworzy najlepsze uporządkowanie dla każdej kwerendy i zapisuje te informacje do struktury danych SeqAlign.

Wyniki wskazują na jakość dopasowania naszych sekwencji. Otrzymywany wynik (raw score) jest normalizowany i porównywany z prawdopodobieństwem uzyskania przypadkowego dopasowania. Jeżeli wynik jest statystycznie istotny, jest wypisywany.

Podsumowanie

Bez BLASTa bioinformatyka miałaby trudności w rozwiązywaniu niektórych problemów. Jak widać jest to algorytm niezbędny do pracy z sekwencjami aminokwasów lub nukleotydów. Jednak nie ma róży bez kolców. Cechuje go też kilka wad, m.in.:

- algorytm zakłada losowość sekwencji,
- dla krótkich sekwencji (ok 20 znakowych) dostajemy niedokładne wyniki,
- zakłada, że całość bazy można reprezentować jako pojedynczą długą sekwencję.

Być może w przyszłości pojawi się jakieś doskonalsze narzędzie, ale na chwilę obecną się na to nie zapowiada.

Bibliografia (nie wymieniona w tekście):

1. *http://www.uwm.edu.pl/bioinfo/dydaktyka/ncbi/BLAST_handbook.pdf* [dostęp 20.02.2015]
2. *http://blast.ncbi.nlm.nih.gov/Blast.cgi* [dostęp 22.02.2015]

Cichy Kącik modelarza

Modelowanie pozwala nam na przedstawienie rzeczywistości w uproszczony, przystępny sposób pozwalający na obserwację ciekawych tendencji. Jest szczególnie ważne w biologii. W pierwszym numerze model będzie stosunkowo zabawny...

Symulacja modelu matematycznego apokalipsy zombie

Rafał Klimek

W ostatnich latach motyw zombie stał się bardzo popularny zarówno w filmie (np. *World War Z*), serialach telewizyjnych (np. *The Walking Dead*), jak i grach komputerowych (np. Dying Light), dlaczego więc nie spojrzeć na tematykę pod kątem naukowym? Artykuł ten opisuje jak zaprezentować apokalipsę zombie za pomocą modelu matematycznego.

Czym tak na prawdę są zombie? Powszechnie są rozumiane jako ożywione ludzkie zwłoki, żywiące się ciałem człowieka. Historie o nich pochodzą od afroamerykańskich wierzeń Voodoo, gdzie zombie zostają wskrzeszone i kontrolowane przez potężnego czarnoksiężnika. Mówi się, że czarnoksiężnik używa proszku zombie do zombifikacji. Proszek ten zawiera silną neurotoksynę, która tymczasowo paraliżuje układ nerwowy człowieka i wprowadza w stan hibernacji. Główne organy takie jak serce czy płuca minimalizują swoje procesy życiowe. Powoduje to niedotlenienie i uszkodzenie mózgu, przez co następuje przemiana człowieka w zombie. Istota ta, ze względu na brak własnej woli, może być bez trudu sterowana przez osobę obdarzoną wiedzą ezoteryczną.

Współczesne zombie różnią się od swoich odpowiedników w voodoo. Film *Noc żywych trupów* z 1968 roku ustandaryzował późniejsze występowania zombie we współczesnej kulturze. Od tej pory stworzenia te są ukazywane jako bezmózgie potwory nieczujące bólu i mające niepohamowany apetyt na ludzkie mięso. Powstają one w wyniku ugryzienia człowieka przez innego nieumarłego. Sensem ich

istnienia jest zabić, zarazić i pojeść sobie. Nieumarli poruszają się w małych, nieregularnych stadach. Można dostrzec na ich ciałach sygnały rozkładu tkanek czy otwartych ran. Zombie często są powodem apokalipsy, gdzie ludzka cywilizacja ginie w wyniku plagi nieumarłych

W poniższym artykule zostanie przedstawiony model matematyczny symulacji plagi zombie występującej w pewnej populacji podatnej na infekcję. W założeniach eksperymentu przyjęto 3 klasy populacji:

- Żywi (S) – podatni na zombifykację
- Zombie (Z)
- Martwi (R)

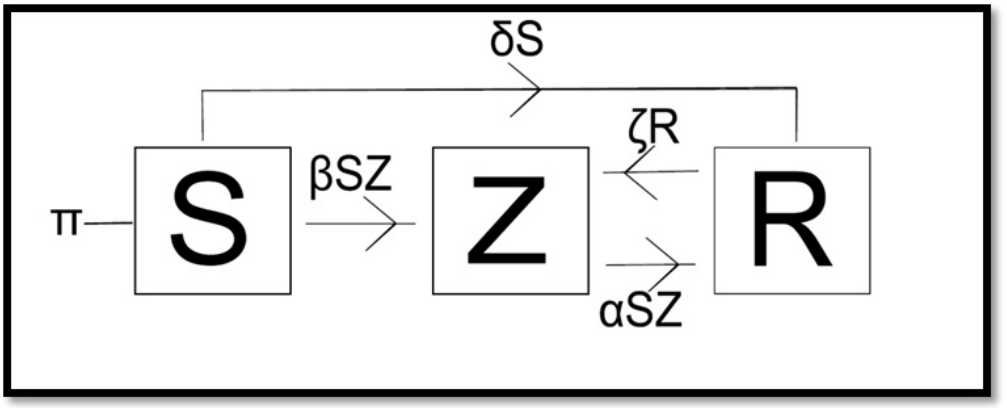
Przyjęto występowanie następujących zjawisk, które zostaną opisane za pomocą parametrów:

- Żywi mogą umrzeć z przyczyn naturalnych - parametr δ.
- Żywi mogą przemienić się w zombie - parametr β.
- Przyjmujemy, że mogą przybyć nowe żywe osoby spoza symulacji – parametr Π.
- Zmarli mogą zostać ożywieni do postaci zombie - parametr ζ.
- Zombie mogą zostań unicestwieni poprzez zniszczenie mózgu – parametr α.

Po uwzględnieniu wszystkich zjawisk otrzymujemy równania zmian liczebności populacji w czasie jednej iteracji:

$$dS = \prod - \beta \cdot S(i) \cdot Z(i) - \delta \cdot S(i)$$
$$dZ = \beta \cdot S(i) \cdot Z(i) + \zeta \cdot R(i) - \alpha \cdot S(i) \cdot Z(i)$$
$$dR = \delta \cdot S(i) + \alpha \cdot S(i) \cdot Z(i) - \zeta \cdot R(i)$$

Schemat modelu wygląda następująco:



Założono następujące wartości parametrów:

$$\alpha = 0{,}0005 \left[\frac{1}{dzien}\right]$$

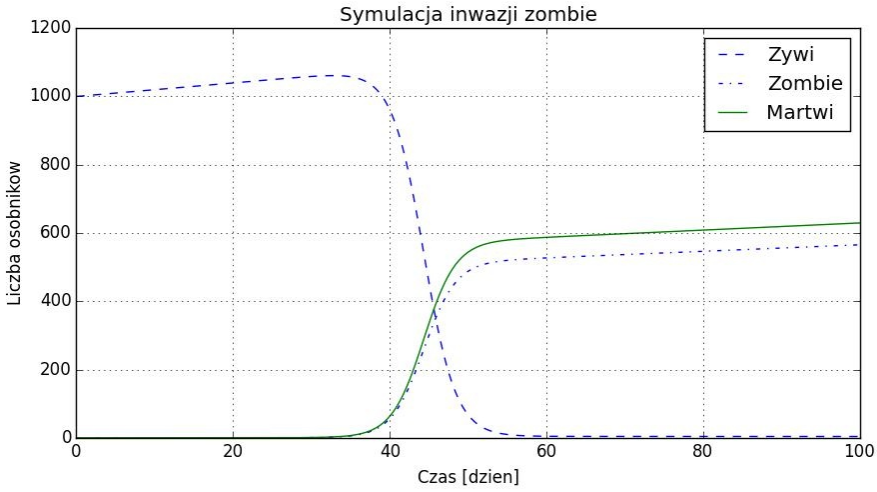
$$\delta = 0{,}00001 \left[\frac{1}{dzien}\right]$$

$$\beta = 0{,}00095 \left[\frac{1}{dzien}\right]$$

$$\zeta = 0{,}00001 \left[\frac{1}{dzien}\right]$$

$$\Pi = 2$$

Wykres symulacji wygląda następująco:



Liczba osobników żywych na początku symulacji była stała, wzrastała dzięki współczynnikowi urodzeń. W początku czterdziestego dnia następuje załamanie, kiedy to zaobserwować można gwałtowny wzrost zombie oraz trupów. Przed 60 dniem populacja żywych przestaje istnieć i obserwujemy tylko powolny spadek pozostałych klas na skutek dostarczania do populacji żywych osób z zewnątrz.

Bibliografia:

Munz P. i in., “*When zombies attack!: Mathematical modelling of an outbreak of zombie infection*”

Kod Apokalipsy (Python):

```
import matplotlib.pyplot as plt

def simulation_zombie():
    pi = 5
    alfa = 0.005
    beta = 0.0095
    gamma = 0.0001
    delta = 0.0001
    S = 1000
    Z = 0
    R = 0
    time = 0
    time_end = 10
    interval = 0.01
    tab_S = []
    tab_Z = []
    tab_R = []
    tab_time = []
    while time < time_end:
        S += (pi - beta * S * Z - delta * S) * interval
        Z += (beta * S * Z + gamma * R - alfa * S * Z) * interval
        R += (delta * S + alfa * S * Z - gamma * R) * interval

        tab_S.append(S)
        tab_Z.append(Z)
        tab_R.append(R)
        tab_time.append(time)
        time += interval

    fig = plt.figure('Wykres symulacji', figsize=(10, 5), dpi=100,\
     facecolor='white')

    plt.plot(tab_S, label='Zywi')
    plt.plot(tab_Z, label='Zombie')
    plt.plot(tab_R, label='Martwi')
    plt.xlabel('Czas')
    plt.ylabel('Liczba osobnikow')
    plt.title('Symulacja inwazji zombie')
    plt.grid(True)
    plt.legend()
    plt.show()

    print(tab_S)
    print(tab_Z)
    print(tab_R)

if __name__ == "__main__":
    simulation_zombie()
```

Artykuł Gościa

Drodzy Członkowie Kół Naukowych, Pracownicy Naukowi i wszyscy Studenci. To miejsce jest dla Was. Jako członkowie najmłodszego koła na Uniwersytecie, dziękujemy za dzielenie się z nami Waszym bezcennym doświadczeniem.

Symulacje stochastyczne, czyli w poszukiwaniu różnorodności

Jan Majta

Na przestrzeni lat opracowano niezliczoną ilość metod obliczeniowych opisujących różne procesy biologiczne. Opracowano formuły pozwalające obliczyć zmiany stężeń substancji w czasie czy poziom ekspresji genów. Położyło to podwaliny pod wiele działów biostatystyki czy bioinformatyki.

Główną wadą metod analitycznych jest fakt, że nie biorą pod uwagę biologicznej zmienności o ile nie jest uwzględniona w zestawie parametrów wzoru matematycznego. Takie podejście zaniedbuje różnorodność pomiędzy eksperymentami czy unikalne zachowania pojedynczych komórek.

Dobrym sposobem radzenia sobie z tym uproszczeniem w celu, na przykład określania wartości danej cechy dla konkretnych elementów zbioru jest stosowanie algorytmów stochastycznych do symulowania procesów biologicznych (ang. *Stochastic Simulations Algorithms* - SSA). Ta metoda pozwala wziąć pod uwagę jednostkową różnorodność między konkretnymi czy kolejnymi komórkami. Ponadto pozwala prześledzić zmiany wartości cech, które są wyraźnie zmienne w czasie. SSA znajdują zastosowanie w prowadzeniu badań dotyczących pojedynczych komórek (ang. single cell - SC). Wartą przybliżenia jest implementacja SSA w języku Python - pakiet *StochPy* rozwijany od kilku lat na Uniwersytecie w Amsterdamie w oparciu o algorytm Gillespiego.

StochPy powstał jako narzędzie do dwojakiego wykorzystania. Pozwala zarówno na pracę interaktywną z poziomu konsoli Pythona, jak również implementacje w większe skrypty. Filozofia stawiająca na maksymalne uproszczenie pracy interaktywnej sprawia jednak, że pewne operacje wymagają głębszego wczytania się w na szczęście przejrzyście napisany i samodokumentujący się kod pakietu.

Mechanizm działania algorytmu stochastycznego opiera się na kilku kluczowych krokach. W pierwszym załadowane zostają parametry symulacji, takie jak startowe stężenia wszystkich reagentów czy częstotliwości syntezy i degradacji. Kolejno następuje krok losowy, podczas którego losowane jest zdarzenie, które zajdzie oraz interwał czasowy, w którym ono zajdzie. Prawdopodobieństwo zajścia danej reakcji obliczane jest w oparciu o podane na wejściu częstotliwości zachodzenia zdarzeń. Następnie przeliczane są nowe ilości reagentów i procedura jest powtarzana. Pakiet StochPy zawiera wbudowane podstawowe modele, takie jak prosta synteza - degradacja czy model ekspresji genów. Co ważne, modele są jasno i prosto zdefiniowane w plikach tekstowych, co umożliwia wygodne ich modyfikowanie i tworzenie własnych układów symulacyjnych.

Dla początkującego użytkownika szczególnie przydatne może okazać się zawarte w dokumentacji pakietu zestawienie przykładowych zastosowań różnych modeli. Poniżej pokrótce opiszę wykorzystanie modelu syntezy mRNA w analizie szumu ekspresji genów.

Jak wiadomo, synteza oraz degradacja mRNA zależy od wielu czynników, które można upraszczając sprowadzić do następujących parametrów: aktywacja i dezaktywacja chromatyny oraz synteza i degradacja mRNA. Ograniczenie metod analitycznych, które również bazują na tych parametrach, to fakt uśredniania populacji komórek. Aby uzyskać informacje o ekspresji konkretnych komórek należy przeprowadzić symulację. W prowadzonych przeze mnie analizach, celem było porównanie szumu ekspresji genów w zależności od poziomu stałej degradacji. Modulacja tego parametru odpowiadała zwiększonej lub zmniejszonej ekspresji interferującego RNA w komórce.

W oparciu o dane literaturowe opracowano zestaw parametrów ekspresji dla transkryptomu ludzkiego. Następnie przeprowadzono dwie serie symulacji dla bazowej oraz zwiększonej częstotliwości degradacji mRNA. Takie symulacje pozwoliły na określenie w jakich warunkach zmiana poziomu mRNA wpływa na międzykomórkowy szum ekspresji genów.

Bibliografia

1. Maarleveld T.R. i in., 2013, „*StochPy: a comprehensive, user-friendly tool for simulating stochastic biological processes*”, PLoS One
2. *http://stochpy.sourceforge.net/*

Nasze zmagania

W niniejszej zakładce przedstawiać będziemy wyniki naszych badań. Chwilowo nie mamy się czym chwalić, ale już wkrótce będziemy kończyć nasze prace licencjackie. Co więcej projekty realizowane przez nasze Koło wejdą w decydujące fazy Mamy nadzieję, że w następnym numerze przedstawimy Wam, drodzy Czytelnicy rezultaty naszych wysiłków.

Poszerzamy horyzonty

Zachęcamy do wzięcia udziału w następujących kursach e-learningowych, prowadzonych przez University of California w San Diego, zamieszczonych na platformie Coursera:

1. *Finding Hidden Messages in DNA* (start w kwietniu)
2. *Assembling Genomes and Sequencing Antibiotics* (rozpoczęcie w maju)
3. *Comparing Genes, Proteins, and Genomes* (początek w lipcu)

https://www.coursera.org//ucsd

Sprawozdanie z działań Koła

Paulina Kania, Karolina Cibor

Koło Bioinformatyki BIT powstało w czerwcu 2014 roku i liczy sobie już ponad 30 członków. W naszym Kole każdy zainteresowany bioinformatyką, biologią obliczeniową, czy biocybernetyką ma możliwość zdobycia doświadczenia i nowych umiejętności, poszerzenia swojej wiedzy oraz poznania wielu ciekawych ludzi.

Seminarium inauguracyjne odbyło się 20 listopada. W jego trakcie zostały wstępnie przybliżone struktura Koła oraz plan działania i cele na nadchodzący rok. Następnie dr n. med. Monika Piwowar w swoim wystąpieniu w sposób ciekawy i zrozumiały zarówno dla infomatyków, jak i osób z szerszą wiedzą biologiczną, przedstawiła związek bioinformatyki z medycyną. Seminarium zostało przyjęte z zainteresowaniem i entuzjazmem.

Pierwsze miesiące działalności Koła skupiały się wokół organizacji seminariów i warsztatów, których tematem były narzędzia i aplikacje używane na co dzień przez bioinformatyków. Głównymi zagadnienieniami poruszanymi na tych spotkaniach były modelowanie biocząsteczek oraz projektowanie leków.

Warsztaty

Seminaria

Obecnie Koło poszerza swoją działalność i prócz przygotowania kolejnej serii seminariów i warsztatów, współorganizuje konferencję "Człowiek Zalogowany" oraz wraz z Kołem Matematyków Studentów UJ im. prof. Stanisława Zaremby i Kołem Studentów Informatyki UJ organizujemy konferencję Liczby-Komputery-Życie, która odbędzie się w dniach 15-17 maja 2015. Ponadto niebawem rozpoczniemy nasz pierwszy projekt naukowy: wykrywanie duplikacji genów w genomie. Jego celem jest stworzenie szybkiego algorytmu wyznaczającego geny powstałe w wyniku duplikacji pochodzące od genu badanego. Aby to osiągnąć wykorzystane zostaną ukryte modele Markowa (ang. hidden Markov models, HMM) z webowym interfejsem graficznym napisanym w Django. Projekt powstaje we współpracy z Zespołem Ekologi Molekularnej i Behawioralnej z Instytutu Nauk o Środowisku UJ.

Więcej informacji znajdziecie na stronie www.bit.ii.uj.edu.pl oraz na profilu Koła na Facebooku (www.facebook.com/koloBITUJ).

https://www.facebook.com/LiczbyKomputeryZycie

Słowniczek dla Biologów

Abstrakcyjny typ danych - formalny opis własności i operacji na danych. Przykładem bytu abstrakcyjnego jest owoc, w odróżnieniu od konkretnej gruszki, która spadła z drzewa rosnącego na ściśle zdefiniowanych współrzędnych geograficznych, w ściśle określonej chwili, o kolorze z dokładnie określoną długością fali elektromagnetycznej itd. (patrz klasa)

Dziedziczenie - uwspólnianie właściwości różnych klas; klasa A dziedzicząca po klasie B zawiera wszystkie atrybuty i zachowania klasy B, jednoocześnie posiadając swoje specyficzne właściwości. Na przykład, jabłko dziedziczy po owocu atrybuty takie jak słodkość, soczystość oraz zachowania typu dojrzewanie, ale posiada specyficzną dla siebie szerokość szupinki, której nie posiada obiekt klasy banan również dziedziczącej po owocu.

Instancja - pojedyncze wystąpienie niezależnego bytu zgodnego z danym wzorcem. Przykładowo, obiektem klasy samochód będzie pojazd o numerze rejestracyjnym QWE-321.

Interfejs - definicja abstrakcyjnego typu posiadającego jedynie operacje, a nie posiadającego danych.

Język naturalny - język używany przez ludzi do komunikacji między sobą; używa się tego pojęcia w celu odróżnienia tego języka od języków formalnych, jakimi są np. języki programowania.

Konstruktor - specjalna metoda danej klasy, wywoływana podczas tworzenia jej instancji.

Klasa - definicja danego obiektu, swego rodzaju przepis lub instrukcja obsługi, zawierająca wszystkie jego cechy (atrybuty), ograniczenia oraz zachowania (metody). Na przykład, obiekt klasy banan posiada atrybuty: wiek, kolor, czy krzywizna, wyróżniające go spośród innych obiektów. Za metodę można uznać na przykład polecenie "dojrzewaj", które jako argument przyjmuje etylen i powoduje zmianę koloru.

Lista - struktura danych służąca do przechowywania obiektów sekwencyjnych

Metoda - elemtent składowy klasy, wykonujący działania na obiektach danej klasy, lub klasy spokrewnionej (patrz dziedziczenie).

Moduł - również pakiet, będący wyodrębnionym tworem zawierającym między innymi treści funkcji, zmienne oraz stałe.

Open Source - wolne oprogramowanie, czyli takie, które może być uruchamiane, kopiowane, rozpowszechniane, analizowane oraz zmieniane i poprawiane przez użytkowników.

Operacja - czynność wykonywana na danym przedmiocie.

Rzutowanie - przekształcanie jednego typu zmiennej w inny, na przykład z liczby zmiennoprzecinkowej w liczbę całkowitą.

String - struktura danych, łańcuch znaków.

Uczenie maszynowe - jest to dziedzina nauki zajmująca się problematyką sztucznej inteligencji; jej głównym celem jest stworzenie automatycznego systemu potrafiącego doskonalić się przy pomocy zgromadzonego doświadczenia (czyli danych) i nabywania na tej podstawie nowej wiedzy

Uczenie nadzorowane - rodzaj uczenia maszynowego, które zakłada obecność ludzkiego nadzoru nad tworzeniem funkcji odwzorowującej wejście systemu na jego wyjście.

Uczenie nienadzorowane - rodzaj uczenia maszynowego, które zakłada brak obecności ludzkiego nadzoru nad tworzeniem funkcji odwzorowującej wejście systemu na jego wyjście.

Słowniczek dla Informatyków

Aminokwasy kanoniczne - dwadzieścia jednostek budulcowych białek i peptydów zakodowanych w RNA.

Aminokwasy niekanoniczne - aminokwasy wchodzące w skład białek, ale nie uwzględniane w kodzie genetycznym, np. selenocytozyna i pirolizyna.

DNA - kwas deoksyrybonukleinowy, podstawowy nośnik informacji w komórkach. W jego skład wchodzą nukleotydy, złożone między innymi z zasad azotowych: adeniny, tyminy, guaniny i cytozyny.

Delecja - rodzaj mutacji, usunięcie jednego lub więcej nukleotydów, razem z insercją zwana InDelem.

Ezoteryka – wiedza dostępna jedynie dla osób uprzywilejowanych, doświadczonych lub takich, które przeszły inicjację.

Format fasta - format zapisu sekwencji biologicznych. Zapis składa się z nagłówka rozpoczętego znakiem '>' i sekwencją w nowej linii.

Format GenBank - format zapisu sekwencji biologicznych. Oprócz sekwencji zawiera wiele użytecznych informacji.

Gen - podstawowa jednostka dziedziczności zapisana w DNA, determinuje powstanie cząsteczki białka lub RNA.

Gpz - jednostka długości kwasu nukleinowego, oznaczająca giga pary zasad.

Hibernacja – fizjologiczny stan organizmu, o charakterze przystosowawczym, polegający na wyłączeniu termoregulacji, znacznym spowolnieniu procesów życiowych i obniżeniu temperatury ciała, zwiększający tolerancję organizmu wobec niesprzyjających warunków środowiskowych.

Insercja - rodzaj mutacji, wstawienie jednego lub więcej nukleotydów, razem z delecją zwana InDelem.

Kod genetyczny - reguła mówiąca, że informacja zawarta w DNA i RNA zawiera informacje o sekwencji białek. Trzy kolejne nukleotydy kodują aminokwas lub przerywają translację.

Mutacja - nagłe zmiany materiału genetycznego, podlegające dziedziczeniu. Zalicza się do nich między innymi delecje, insercje, mutacje punktowe.

Mutacja punktowa - mutacja polegająca na zamianie jednego nukleotydu innym.

Neurotoksyny − rodzaj toksyn działających na układ nerwowy.

Nić komplementarna - nić powstała na matrycy DNA, poprzez przyłączanie tymin na miejsce adenin matrycy oraz guanin w miejsce cytozyn (i odwrotnie).

Odwrotna transkrypcja - biosynteza DNA na matrycy RNA.

Phred quality - Q = - log (P), gdzie P jest prawdopodobieństwem błędnego przypisania nukleotydu do danej pozycji DNA lub RNA. Q "koduje się" symbolami z kodu ASCII.

RNA - kwas rybonukleinowy, produkt translacji DNA, różniący się między innymi zamianą tymin charakterystycznych dla DNA uracylami.

Sekwencja kodująca - część sekwencji kwasu nukleinowego zawierająca informacje
o białku

Transkrypcja - proces "przepisywania" DNA na RNA, biosynteza nici RNA

Translacja - proces "tłumaczenia" RNA na białko.