SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Projekt iz predmeta Bioinformatika 2

Poravnanje parova sljedova korištenjem skrivenog Markovljevog modela

Studenti: Ivan Furač i Jana Martinović

Mentor: izv. prof. dr. sc. Mirjana Domazet-Lošo

Sadržaj

Uvod

Cilj ovoga projekta bio je implementirati model za poravnanje parova sljedova temeljen na teoriji skrivenih Markovljevih modela. Poravnanje bioloških sljedova u svrhu utvrđivanja njihove sličnosti, bilo sljedova nukleotida ili sljedova aminokiselina, jedan je od temeljnih zadataka područja bioinformatike. Postoji niz egzaktnih algoritama koji je do sada razvijen za rješavanje ovog zadatka, međutim većina tih algoritama susreće se s problemom velike složenosti koja dolazi do izražaja kada se poravnavaju sljedovi veće duljine, na primjer duljine više tisuća nukleotida ili aminokiselina. Poravnanje temeljeno na skrivenim Markovljevim modelima koristi vjerojatnosti zbog čega možda neće producirati najoptimalnije poravnanje, međutim ima puno manju složenost u odnosu na ostale algoritme. Model je najprije potrebno istrenirati nad skupom unaprijed poravnatih parova, za što se koristi Baum-Welch algoritam. Treniranjem modela pronalaze se optimalni parametri Markovljevog modela. Istrenirani model se zatim može koristiti za poravnavanje dva slijeda, za što se koristi Viterbijev algoritam. Kako bi se odredila točnost dobivenih poravnanja, rezultati dobiveni ovom metodom uspoređuju se s rezultatima koji su dobiveni nekim od egzaktnih algoritama, za što se u ovom projektu koristi Needleman-Wunsch algoritam.

Skriveni Markovljevi modeli

Skriveni Markovljev model sastoji se od stanja te opservacija koje pojedino stanje emitira. Stanja nije moguće izravno opaziti, odnosno ona su skrivena. Umjesto njih opaža se niz opservacija. Parametri skrivenog Markovljevog modela su sljedeći:

- Pi – matrica početnih vjerojatnosti za svako stanje
- A – matrica vjerojatnosti prijelaza iz jedno stanje u drugo
- E – matrica vjerojatnosti emisije pojedinih simbola u svakom stanju

Model koji opisuje poravnanje dva slijeda sastoji se od sljedećih stanja:

- M (match/mismatch) – stanje u kojem se emitiraju simboli iz oba slijeda
- Ix (insert x) – stanje u kojem se emitira samo simbol iz prvog slijeda, a za drugi slijed se emitira praznina (engl. gap)
- Iy (insert y) – stanje u kojem se emitira samo simbol iz drugog slijeda, a za prvi slijed se emitira praznina

U stanju M moguće su opseravcije parova simbola iz skupa {A, C, G, T}, odnosno postoji ukupno 16 mogućnosti opservacija. U stanjima Ix i Iy emitira se samo po jedan simbol iz prethodno navedenog skupa, dakle moguće su samo 4 opservacije u svakom stanju.

Struktura projekta

Cjeloviti kod projekta može se pronaći na GitLab-u, na poveznici navedenoj na kraju dokumentacije [1]. Programski se kod može podijeliti u 4 cjeline koje su napisane u sljedećim programskim jezicima:

- Pretprocesiranje podataka – Python
- Treniranje skrivenog Markovljevog modela – C++
- Generiranje poravnanja – C++
- Bodovanje poravnanja – Python

Pretprocesiranje podataka

Podatci koji su korišteni u ovom projektu preuzeti su s online baze podataka sljedova HIV-a američkog Nacionalnog laboratorija u Los Alamosu [2]. Preuzeti podatci sadrže velik broj sljedova HIV-a koji su višestruko poravnati (engl. multiple alignment).

Dio koda koji se koristi za pretprocesiranje podataka nalazi se u mapi ''hmm_scripts_data''. Preuzeti podatci nalaze se u datoteci ''hiv_alignment2.fasta'' koja sadrži 2440 višestruko poravnatih sljedova. Python skripta ''preprocessing.py'' služi za generiranje skupova parova za treniranje i testiranje, kao i estimacije matrica inicijalnih parametara Pi*, A* i E*.

Generiranje skupova za treniranje i testiranje vrši se na sljedeći način:

- iz datoteke ''hiv_alignment2.fasta'' uzimamo 20 nasumičnih sljedova
- izbacimo one sljedove koji sadrže specijalne znakove (svi simboli koji nisu iz skupa {A, C, G, T, -}
- generiramo sve moguće kombinacije parova
    - ⅔ odvojimo za treniranje, ⅓ za testiranje
- na skupu za treniranje napravi se sljedeće:
    - poravnanja svakog para pretvorimo iz višestrukog u jednostruko
        - funkcija remove_dash_dash(first, second) vraća parove takve da su izbačena sva pojavljivanja ''-'', ''-'' na istom indeksu

        - funkcija remove_gapx_gapy(first, second) vraća parove takve da su (''P'', ''-''), (''-'', ''Q'') zamijenjena s (''P'', ''Q'')
    - spremimo parove u direktorij ''training_data'', svaki u zasebnu datoteku ''training_no_[index]''
- na skupu za testiranje napravi se sljedeće (funkcija generate_testing_data):
    - poravnanja svakog para pretvorimo iz višestrukog u jednostruko
    - spremimo parove u direktorij ''testing_data_aligned'', svaki u zasebnu datoteku ''aligned_no_[index]''
    - uklonimo svaku pojavu znaka ''-''
    - spremimo parove u direktorij ''testing_data'', svaki u zasebnu datoteku ''test_no_[index]''

Generiranje inicijalnih parametara vrši se na sljedeći način:

- funkcija estimate_pi(pairs) prima listu parova i vraća rječnik čiji su ključevi stanja, a vrijednosti frekvencije
    - za svaki par treba saznati kojim stanjem započinje te inkrementirati to stanje
    - na kraju treba podijeliti s ukupnim brojem parova
- funkcija estimate_A(pairs) prima listu parova i vraća rječnik čiji su ključevi početna stanja, vrijednosti su novi rječnik čiji su ključevi konačna stanja te vrijednosti frekvencije pojavljivanja
    - za svaki par te za svaki podniz znakova duljine 2 treba saznati iz kojeg se stanja prešlo u koje te inkrementirati taj prijelaz
    - na kraju treba podijeliti s ukupnim brojem prijelaza
- funkcija estimate_E(pairs) prima listu parova i vraća rječnik čiji su ključevi stanja, a vrijednosti su novi rječnik čiji su ključevi emisije te vrijednosti frekvencije pojavljivanja
    - za svaki par treba saznati stanje i očitati emisije te inkrementirati tu emisiju
    - na kraju treba podijeliti s ukupnim brojem emisija

Rezultate ovih triju funkcija spremamo redom u datoteke ''matrix_pi_values'', ''matrix_a_values'', ''matrix_e_values''. Svaka nova vrijednost se zapisuje u novi redak, a pomoćna tablica ključeva nalazi se redom u datotekama ''matrix_pi_keys'', ''matrix_a_keys'' te ''matrix_e_keys''.

Treniranje skrivenog Markovljevog modela

Datoteke s inicijalnim vrijednostima parametara Markovljevog modela nalaze se u mapi ''start_params''. Programski kod za treniranje u ovom direktoriju očekuje 3 datoteke naziva ''Pi'', ''A'' te ''E'' u kojima se nalaze inicijalne vrijednosti. U mapi ''train_pairs'' trebaju se nalaziti datoteke od kojih svaka sadrži po jedan par dva unaprijed poravnata slijeda, a koje su dobivene u prethodnom koraku pretprocesiranja podataka. U ovom se projektu za treniranje koristi 80 parova sljedova. Za treniranje je dovoljno pokrenuti kod u datoteci ''Main_train.cpp''. U toj su datoteci definirane putanje do mapa u kojima su pohranjeni parovi za treniranje te inicijalni parametri Pi*, A* i E*. Datoteka ''Parser.cpp'' sadrži pomoćne funkcije koje se koriste za učitavanje parova sljedova i parametara te njihovo spremanje u prikladne strukture podataka. Prilikom učitavanja parametara koji zapravo predstavljaju vjerojatnosti, vrijednosti koje iznose 0.0 zamjenjuju se sa zanemarivo malim vrijedostima kako kasnije ne bi došlo do računskih pogrešaka. Također, u strukture podataka se pohranjuju logaritmi učitanih vjerojatnosti, s obzirom da bi u suprotnom zbog velikog broja operacija množenja vrijednosti parametara brzo konvergiraju prema nuli. U datoteci ''HMM.cpp'' nalazi se klasa koja predstavlja Markovljev model i koja se koristi za lakše prenošenje svih parametara između različitih funkcija.

Treniranje modela provodi se kroz više iteracija. U svakoj iteraciji obavi se jedan prolazak kroz cijeli skup parova za treniranje, izvršavaju se koraci Baum-Welch algoritma i na temelju njih se ažuriraju parametri Pi, A i E. Cijeli kod za treniranje nalazi se u datoteci ''Baum-Welch.cpp''. U datoteci se nalaze sljedeće funkcije:

- Funkcija baumwelch je glavna funkcija koja se koristi za pokretanje procesa treniranja modela. Funkcija prima sljedeće argumente:
    - pokazivač na objekt tipa HMM
    - vektor parova sljedova
    - broj iteracija algoritma

  Funkcija u svakoj iteraciji poziva funkciju run_iteration, koja predstavlja jednu iteraciju algoritma. Nakon što su izvršene sve iteracije, konačni parametri modela upisuju se u datoteke koje se spremaju u mapu ''trained_params''.

- Funkcija run_iteration prima sljedeće argumente:
    - pokazivač na objekt tipa HMM
    - vektor parova sljedova

  Funkcija obavlja prolazak kroz sve parove sljedova koji se nalaze u skupu parova za treniranje. Za svaki par izvršavaju se sljedeći koraci:

    - računanje unaprijednih vjerojatnosti
    - računanje unazadnih vjerojatnosti
    - računanje parametra gama
    - računanje parametra ksi
    - pohrana izračunatih parametara

  Nakon što je obavljen prolaz kroz sve parove za treniranje, računaju se konačne vrijednosti te se spremaju u objekt tipa HMM, čiji je pokazivač primljen kao argument.

- Funkcija forward prima sljedeće argumente:
    - pokazivač na objekt tipa HMM
    - jedan par sljedova

  Funkcija računa unaprijedne vjerojatnosti te vraća pokazivač na polje u koje su spremljeni izračunati brojevi.

- Funkcija backward prima sljedeće argumente:
    - pokazivač na objekt tipa HMM
    - jedan par sljedova

  Funkcija računa unazadne vjerojatnosti te vraća pokazivač na polje u koje su spremljeni izračunati brojevi.

- Funkcija observatonPossible prima sljedeće argumente:
    - oznaka stanja (0 označava stanje M, 1 stanje Ix te 2 stanje Iy)
    - simbol iz prvog slijeda
    - simbol iz drugog slijeda

  Funkcija vraća informaciju o tome je li u zadanom stanju moguće opaziti primljeni par simbola.

- Funkcija convertToSymbol prima sljedeće argumente:
    - simbol iz prvog slijeda
    - simbol iz drugog slijeda

  Funkcija spaja primljene simbole u jedan zapis koji vraća.

U kodu je zadano da se za treniranje modela koristi 5 iteracija. Pokazalo se da su nakon tih 5 iteracija promjene u parametrima bile zanemarivo male.

Generiranje poravnanja

Nakon što su pronađeni optimalni parametri Markovljevog modela, on se može koristiti za generiranje poravnanja. Za generiranje poravnanja dovoljno je pokrenuti kod u datoteci ''Main_test.cpp''. U datoteci su definirane putanje do mapa u kojima se nalaze tzv. parovi za testiranje, odnosno neporavnati parovi, te putanje do datoteka u kojima se nalaze optimalni parametri koji su rezultat treniranja. Za generiranje poravnanja koristi se Viterbijev algoritam. Kod se nalazi u datoteci ''Viterbi.cpp''. Viterbijev algoritam koristi se za rješavanje problema dekodiranja, odnosno pronalaženja najvjerojatnijeg niza stanja za zadanu opservaciju, odnosno za zadani par sljedova. Najvjerojatniji niz skrivenih stanja pronalazi se korištenjem Viterbijevih matrica u kojima su pohranjene vjerojatnosti da se model za zadani par simbola nalazi u određenom stanju, te matrica optimalnog puta koje pokazuju koje je najvjerojatnije prethodno stanje iz kojeg se dolazi u trenutno stanje, za svaki par simbola iz ulaznih sljedova. S obzirom da model ima 3 stanja, koriste se 3 Viterbijeve matrice te 3 matrice optimalnog puta. Zbog memorijske efikasnosti, u slučaju Viterbijevih matrica dovoljno je spremati samo 2 retka, odnosno trenutni redak i prethodni redak. Nakon što je pronađen najvjerojatniji niz skrivenih stanja, poravnanje sljedova može se pronaći tako da se redom čitaju stanja iz niza te se za stanje M emitiraju oba simbola iz ulaznih sljedova, za stanje Ix emitira se par (simbol iz prvog slijeda, -) te se za stanje Iy emitira par (-, simbol iz drugog slijeda). Poravnati parovi sljedova pohranjuju se u mapu ''alignments''.

Bodovanje poravnanja

Kako bismo mogli ocijeniti ispravnost algoritma valjalo je rezultate usporediti s onima koje daje Needleman Wunsch algoritam. Odlučili smo koristiti Python biblioteku ''minineedle'' koja u sebi ima gotove funkcije za poravnanje algoritmom Needleman Wunsch. Nakon što smo poravnali sve parove za testiranje rezultate smo spremili u direktorij ''needleman'', svaki u zasebnu datoteku ''needleman_no_[index]''.

Kako bismo mogli procijeniti koliko su dobivena poravnanja kvalitetna u odnosu na poravnanja dobivena algoritmom Needleman Wunsch koristili smo algoritam ''bodovanja poravnanja'' (*alignment scoring algorithm)*.

- funkcija get_score(first, second) vraća float vrijednost koja označava broj bodova koje je pojedino poravnanje ostvarilo:
    - za svaku emisiju potrebno je saznati trenutno stanje
    - prvi pogodak (*start_match*), nastavni pogodak (*continue_match*), pogreška (*mismatch*), otvaranje procjepa (*start_gap*), nastavak procjepa (*continue match*)
- u konačni rezultat potrebno je nadodati unaprijed dogovorene (u dvije varijante) nagrade i kazne za navedene rezultate
    - varijanta 1-2-3: continuing the gap (-1), starting the gap (-2), continuing the match (+3), starting the match (+2), mismatch (+1)
    - varijanta 1-5: continuing the gap (-1), starting the gap (-5), continuing the match (+5), starting the match (+5), mismatch (+1)

Ovaj postupak ponavljamo za različita poravnanja:

- poravnanja dobivena korištenjem skrivenog Markovljevog modela (*hmm_aligned*)
- poravnanja dobivena algoritmom Needleman Wunsch (*needleman*)
- poravnanja dobivena pretvaranjem testnih podataka u poravnanje parova sljedova manipulacijom stringova (*testing_data)*

Bodove za sva navedena poravnanja spremamo u datoteku ''scores.tsv'', a prosječne bodove u datoteku ''average_scores.tsv''.

Rezultati

Dobivena poravnanja su smislena, ali nešto niže kvalitete od očekivanog. Model jako rijetko otvara procjepe (gaps) i kada ih otvori oni budu vrlo kratki. Pogreške (mismatches) preferira nad procjepima što odudara od očekivanja i naravno, znatno utječe na bodovanje poravnanja.

Bodovi poravnanja (varijanta 1-2-3) dobivena pomoću HMM su u prosjeku 33% niži od poravnanja dobivena pomoću algoritma NW (37% za varijantu 1-5).

U nastavku slijede konkretni rezultati bodovanja za varijantu 1-2-3 za 40 različitih parova sljedova koji su korišteni za testiranje. U lijevim se stupcima nalaze rezultati bodovanja za algoritam NW (Needleman-Wunsch), a u desnim se stupcima nalaze rezultati za poravnanja dobivena pomoću skrivenog Markovljevog modela.

NW	HMM
59560	35610
59470	34680
59140	36770
65890	48480
67470	36300
58510	37580
64650	49260
60760	34390
59880	37890
60680	34400
59790	35530
64500	53090
59420	36640
58830	48870
65910	41750
61830	36320
64590	51480
65470	41600
59790	36050
65620	36320

NW	HMM
64590	50030
65130	42980
58980	40800
65780	48850
61280	38680
59780	49930
59620	45270
66890	49810
65650	47350
60200	54920
64360	34850
67070	42030
66100	54400
65080	36520
64520	49860
67960	36180
66790	36190
66110	36370
65970	35210
58730	35020

Poveznice

[1] Poveznica na projekt na GitLab-u

https://gitlab.com/SlonFurki/hmm

[2] Poveznica na online bazu podataka sljedova HIV-a

https://www.hiv.lanl.gov/content/sequence/NEWALIGN/align.html.