Απαλλακτική Εργασία 2024

Μάθημα: **Βιοπληροφορική**

Φοιτητές:

Αγγελάκης Κωνσταντίνος Π21001

Λειβαδάρος Παναγιώτης Π21085

Εξάμηνο: **6ο**

Εισαγωγή

Για τις ανάγκες της απαλλακτικής εργασίας του εξαμήνου στο μάθημα της Βιοπληροφορικής, υλοποιήσαμε όλα τα ζητούμενα ερωτήματα που περιείχε η παρούσα άσκηση. Για την τεκμηρίωση του προγράμματος, θα ξεκινήσουμε από την ανάλυση του αρχείου main.py, που φυσικά δεν ανήκει σε κάποιο μεμονωμένο υποερώτημα, και θα συνεχίσουμε με τα υπόλοιπα τρία αρχεία .py, κάθε ένα από τα οποία αντιστοιχεί σε ένα υποερώτημα της εργασίας. Τέλος θα παραθέσουμε τη δομή των φακέλων, ώστε ο αναγνώστης να μπορεί να βρει τα επιθυμητά αρχεία πιο εύκολα.

Αρχείο main.py

Το αρχείο main.py, συντονίζει την διεκπεραίωση των λειτουργιών του προγράμματος, λειτουργώντας ως η ραχοκοκαλιά του, όπως υποδηλώνει και το όνομά του. Ξεκινάμε, λοιπόν, κάνοντας import σε αυτό τα τρία αρχεία του project, κάθε ένα εκ των οποίων υλοποιεί τη λύση για κάθε ένα από τα ερώτημα της άσκησης μέσα σε συναρτήσεις. Έπειτα, ορίζουμε το αλφάβητο και τα τέσσερα αρχικά patterns, όπως μας υποδηλώνει η εκφώνηση, αλλά και τις τιμές επιβράβευσης ή ποινής του δεύτερου ερωτήματος. Να σημειωθεί εδώ πως οι Αριθμοί Μητρώου και των δύο μας τελειώνουν, όπως φαίνεται και στο εξώφυλλο του παρόντος αρχείου σε περιττό αριθμό (Π21001 και Π21085), συνεπώς έχουμε α = 1. Κατόπιν, εκτελούμε το πρώτο ερώτημα της εργασίας δημιουργώντας 50 συμβολοσειρές με τον τρόπο που περιγράφεται στην εκφώνηση (θα αναλυθεί περαιτέρω στην ανάλυση του αρχείου generation.py) και έπειτα αναθέτουμε τις πρώτες 15 στο σύνολο Α, και τις υπόλοιπες 35 στο σύνολο Β. Εκτυπώνουμε τις συμβολοσειρές των δύο συνόλων, μαζί με τον αύξον αριθμό τους. Στη συνέχεια, καλούμε τη συνάρτηση align() του αρχείου alignment.py (κατά την ανάλυση του οποίου θα εξετασθεί περαιτέρω), η οποία, με δεδομένα τις συμβολοσειρές του συνόλου Α και τις τιμές της επιβράβευσης της τοπικής ομοιότητας, της ποινής της τοπικής ανομοιότητας και της ποινής κενού, εκτελεί την διαδικασία της πολλαπλής στοίχισης, πάνω στις εν λόγω συμβολοσειρές. Μόλις η διαδικασία περατωθεί, εκτυπώνουμε τα αποτελέσματα της στοίχισης (στοιχισμένες συμβολοσειρές, μαζί με τον αύξον αριθμό τους). Τέλος, κατασκευάζουμε το Hidden Markov Model Profile βασιζόμενοι στο αλφάβητο μας (συν του συμβόλου της κάτω παύλας «_», το οποίο εσκεμμένα δεν συμπεριλάβαμε στο αρχικό αλφάβητο για λόγους συμφωνίας με την εκφώνηση) και στις στοιχισμένες συμβολοσειρές του συνόλου Α, και το αποθηκεύουμε σε ένα αντικείμενο τύπου HMM (περισσότερα για την κλάση HMM στην ανάλυση του αρχείου hmm.py). Βάση αυτού του προφίλ ρυθμίζουμε τους σχετικούς πίνακες πιθανοτήτων και έπειτα στοιχίζουμε τις συμβολοσειρές του συνόλου Β, υπολογίζοντας φυσικά και τα alignment scores και alignment paths για τι συμβολοσειρές αυτές. Εκτυπώνουμε το γενικό Hidden Markov Model Profile, τη συμβολοσειρά του συνόλου Β μαζί με τον αύξον αριθμό της, και το alignment score και alignment path που της αντιστοιχεί, ώστε το πρόγραμμα λαμβάνει τέλος.

Αρχείο generation.py – υποερώτημα i

Προχωρώντας στο ερώτημα i και το αρχείο που το υλοποιεί, το αρχείο generation.py, έχουμε φτιάξει δύο συναρτήσεις για αυτό το σκοπό. Η πρώτη ως προς το χρόνο κλήσης της είναι η generate_dataset(), η οποία, με όρισμα το αλφάβητο, τα τέσσερα δοσμένα patterns και τον επιθυμητό αριθμό συμβολοσειρών που θέλουμε να παράγουμε (εδώ ισούται με 50 συμβολοσειρές), κατασκευάζει τις συμβολοσειρές αυτές με τον ζητούμενο τρόπο. Αναλυτικότερα, ξεκινάει με το ζητούμενο (α), κατά το οποίο επιλέγει 1 – 3 σύμβολα του αλφάβητου με τυχαίο τρόπο και τα τοποθετεί στην αρχή της καινούργιας συμβολοσειράς. Έπειτα, καλεί τη δεύτερη συνάρτηση του αρχείου, την modify(), μία φορά για κάθε ένα από τα patterns της εκφώνησης. Η εν λόγω συνάρτηση διεκπεραιώνει το ζητούμενο (β), και θα εξεταστεί σε λίγο, ενώ προσθέτει το νέο κομμάτι της συμβολοσειράς στο τέλος της υπάρχουσας. Αφότου η συνάρτηση αυτή επιστρέψει αποτελέσματα για κάθε μία συμβολοσειρά, η αρχική μας συνάρτηση generate_dataset() λαμβάνει την τροποποιημένη συμβολοσειρά ως έχει μέχρι ώρας, και επιλέγει δύο σύμβολα του αλφάβητου με τυχαίο τρόπο και τα τοποθετεί στο τέλος αυτής. Η πλήρως διαμορφωμένη συμβολοσειρά προστίθεται στο σύνολο, μαζί με τις υπόλοιπες. Η διαδικασία αυτή επαναλαμβάνεται όσες φόρες μας υποδεικνύει το τρίτο όρισμα της συνάρτησης, ώστε να έχουμε στο τέλος τον επιθυμητό αριθμό συμβολοσειρών. Τέλος, το σύνολο των συμβολοσειρών επιστρέφεται στη main.

Για κάθε ένα από τα τέσσερα patterns λοιπόν, η συνάρτηση modify() λαμβάνει αυτό το Pattern, Καθώς και μια επαυξημένη εκδοχή του αρχικού αλφάβητου (η οποία περιέχει και το σύμβολο «_»), για να δημιουργήσει το επόμενο κομμάτι της νέας συμβολοσειράς, το οποίο θα προστεθεί στο τέλος αυτής, στη συνάρτηση generate_dataset(), η οποία και κάλεσε την modify(). Η συνάρτηση αυτή, αντικαθιστά 1 – 2 σύμβολα του τρέχοντος pattern σε τυχαίες θέσεις, με κάποιο, τυχαία πάλι επιλεγμένο, σύμβολο του επαυξημένου αλφάβητου. Μέσα σε αυτή τη διαδικασία ελέγχουμε πρώτον, εφόσον επιλεγεί να αλλαχθούν δύο σύμβολα, τη δεύτερη φορά να μην αλλαχθεί το σύμβολο στη θέση που αλλάχθηκε και στην πρώτη επανάληψη (διότι εδώ θα υπήρχε, ουσιαστικά, μία αλλαγή και όχι δύο), και δεύτερον το στοιχείο το οποίο θα αλλαχθεί κάθε φορά να μην αλλαχθεί με τον εαυτό του (διότι εδώ δεν θα γινόταν, ουσιαστικά, αλλαγή, αλλά το σύμβολο στη συγκεκριμένη θέση θα παρέμενε το ίδιο). Έχοντας πλέον τροποποιήσει το δοσμένο pattern, το επιστρέφει στη συνάρτηση generate_dataset, για να προστεθεί στο τέλος της καινούργιας συμβολοσειράς του συνόλου.

Αρχείο alignment.py – υποερώτημα ii

Συνεχίζουμε με το αρχείο alignment.py, το οποίο περιέχει τον κώδικα που υλοποιεί ολόκληρο το δεύτερο ερώτημα της εργασίας. Ξεκινάμε, έτσι, με την πρώτη του κατά σειρά συνάρτηση, την align(), η οποία καλείται από την main. Καταρχάς, ξεκινάμε θέτοντας την μεγαλύτερη σε μήκος συμβολοσειρά του συνόλου Α ως «βάση» για την διαδικασία πολλαπλής στοίχισης, ως δηλαδή τη μοναδική συμβολοσειρά η οποία δεν θα στοιχιστεί η ίδια, αλλά οι υπόλοιπες βάση αυτής. Κατόπιν, εκτελούμε τη διαδικασία πολλαπλής στοίχισης με τη συνάρτηση calculate_grid(την οποία θα αναλύσουμε παρακάτω), κάθε φορά μεταξύ της επόμενης συμβολοσειράς του συνόλου Α και της συμβολοσειράς – βάσης που αναφέραμε προηγουμένως, για κάθε μία συμβολοσειρά του συνόλου Α, πέρα από αυτή που θέσαμε ως βάση. Προσθέτουμε το αποτέλεσμα της κάθε στοίχισης στο σύνολο των στοιχισμένων συμβολοσειρών (όπως και τη συμβολοσειρά βάση), και το επιστρέφουμε στη main, για να εκτυπωθεί και να χρησιμοποιηθεί στο υπόλοιπο πρόγραμμα.

Η επόμενη και ιδιαίτερα σημαντική για το πρόγραμμα συνάρτηση είναι η calculate grid, η οποία υπολογίζει τις τιμές των κελιών του πλέγματος, βάση του αλγορίθμου των Needleman – Wunsch. Αρχίζουμε αρχικοποιόντας με μηδενικά τον πίνακα μεγέθους κατά μία μονάδα μεγαλύτερο από το μήκος της πρώτης προς στοίχιση συμβολοσειράς στη μία διάσταση, και κατά μία μονάδα μεγαλύτερο από το μήκος της δεύτερης προς στοίχισης συμβολοσειράς στην άλλη διάσταση. Έπειτα υπολογίζουμε τις τιμές των κελιών της πρώτης γραμμής και της πρώτης στήλης, όλα εκ των οποίων υπολογίζονται με το gap penalty, αφού πρόκειται για οριζόντιες και κάθετες, αντίστοιχα, κινήσεις. Εδώ, να σημειωθεί ότι η επιβράβευση τοπικής ομοιότητας, η ποινή τοπικής ανομοιότητας και η ποινή κενού, είναι αποθηκευμένες με αυτή τη σειρά σε μια python list με όνομα scores. Μετά από αυτό, συνεχίζει υπολογίζοντας τα υπόλοιπα, «εσωτερικά» κελιά του πίνακα, βάση του αλγορίθμου και των δοσμένων τιμών του πίνακα scores. Εν συνεχεία, καλεί την τελευταία συνάρτηση του παρόντος αρχείου, την find_path (που θα αναλύσουμε πιο κάτω), για να βρει το βέλτιστο μονοπάτι διάταξης των δύο συμβολοσειρών. Τέλος, παίρνει αυτό το μονοπάτι, και υπολογίζει την συμβολοσειρά που σχηματίζει (που αναπαριστά δηλαδή), ως εξής: Αν η κίνηση που γίνεται στο μονοπάτι είναι διαγώνια, τότε εκχώρησε στη συμβολοσειρά το σύμβολο της δεύτερης από τις δύο στοιχιζόμενες συμβολοσειρές που βρίσκεται στην ακριβώς προηγούμενη θέση (δηλαδή στη θέση της συμβολοσειράς πριν γίνει η διαγώνια μετάβαση). Διαφορετικά, αν γίνει κάθετη ή οριζόντια μετάβαση, πρόσθεσε το σύμβολο της κάτω παύλας «_». Επιστρέφει την στοιχισμένη συμβολοσειρά που προκύπτει.

Σκοπός, τώρα, της τελευταίας συνάρτησης του αρχείου που εξετάζουμε, της find_path(), είναι να βρει το βέλτιστο μονοπάτι για στοίχιση των δύο δοσμένων κάθε φορά συμβολοσειρών, τις οποίες λαμβάνει ως όρισμα, μαζί με το πλέγμα στοίχισής τους, και τον πίνακα scores που αναλύσαμε προηγουμένως. Όπως ορίζει ο αλγόριθμος των Needleman – Wunsch, ξεκινάμε από το τελευταίο κελί του πίνακα, του οποίου της συντεταγμένες προσθέτουμε στο μονοπάτι ως

ζεύγος, και συνεχίζουμε προς τα υπόλοιπα. Για κάθε θέση που έχει ήδη επιλεγεί ως μέρος του βέλτιστου μονοπατιού, ελέγχουμε αν έγινε διαγώνια κίνηση (δηλαδή αν η τιμή της θέσης του πίνακα στις τρέχουσες συντεταγμένες ισούται με την τιμή της προηγούμενης διαγώνιας θέσης, συν είτε την επιβράβευση τοπικής ομοιότητας ή την ποινή τοπικής ανομοιότητας), ή αν έγινε κάθετη κίνηση (δηλαδή η αν η τιμή της θέσης του πίνακα στις τρέχουσες συντεταγμένες ισούται με την τιμή της προηγούμενης προς τα πάνω θέσης συν την ποινή κενού), ή αν έγινε οριζόντια κίνηση (περίπτωση αντίστοιχη με την κάθετη κίνηση, συμπεριλαμβάνεται ως τελική συνθήκη τύπου else). Σε κάθε περίπτωση πραγματοποιούμε τους απαραίτητους ελέγχους ώστε να μην βγούμε εκτός των ορίων του πίνακα. Ανάλογα με το ποια επιλογή κάνει ο αλγόριθμος, η κατάλληλη από τις δύο συντεταγμένες μειώνεται κατά μία μονάδα (ή και οι δύο στην περίπτωση της διαγώνιας κίνησης), και οι νέες τους τιμές αποθηκεύονται στο μονοπάτι ως ζεύγος. Αντιστρέφουμε το μονοπάτι, ώστε να έχει μια ροή από την αρχή προς το τέλος (δηλαδή από πάνω - αριστερά προς τα κάτω - δεξιά), και το επιστρέφουμε στη προηγούμενη συνάρτηση, την calculate_grid().

Αρχείο hmm.py – υποερώτημα iii

Τελευταίο και ίσως σημαντικότερο αρχείο της εργασία είναι το hmm.py, το οποίο υλοποιεί ολόκληρο το τρίτο ερώτημα της εργασίας. Κατασκευάζει το Hidden Markov Model Profile, ρυθμίζει του πίνακες μετάβασης και τους πίνακες εκπομπής, και υπολογίζει τα alignment scores και alignment paths για τις ακολουθίες του συνόλου B. Σε αυτό, λοιπόν, το αρχείο έχουμε υλοποιήσει μία κλάση, την HMM, με συναρτήσεις οι οποίες εξυπηρετούν τις λειτουργίες αυτές.

Συγκεκριμένα, ξεκινάμε με τα attributes της κλάσης τα οποία είναι το alphabet, που είναι το αλφάβητο των συμβόλων που μπορούν να εμφανιστούν στις ακολουθίες, τα states, που είναι οι καταστάσεις του μοντέλου HMM (εδώ είναι δύο: M - Match και D - Delete), το aligned_dataset, που είναι το σύνολο των στοιχισμένων συμβολοσειρών που δίνεται ως είσοδος, τα transition_probs, που είναι οι πιθανότητες μετάβασης μεταξύ των καταστάσεων, και τέλος τα emission_probs, που είναι οι πιθανότητες εκπομπής συμβόλων από τις καταστάσεις.

Έχουμε, μετά, τον constructor της κλάσης, ο οποίος αρχικοποιεί τα attributes alphabet και aligned_dataset της κλάσης με τις τιμές που του δίνουμε εμείς ως όρισμα από τη main (το aligned_dataset παίρνει ως τιμή την λίστα από στοιχισμένες συμβολοσειρές του συνόλου A). Κατόπιν, υπολογίζει τις πιθανότητες μετάβασης και εκπομπής καλώντας τη συνάρτηση calculate_probabilities() (που θα εξετασθεί αμέσως μετά), αποθηκεύοντας τα αποτελέσματά της στα δύο ορίσματα της κλάσης.

Η πρώτη μέθοδος αυτού του αρχείου, είναι, όπως αναφέραμε η calculate_probabilities(), η οποία υπολογίζει της πιθανότητες μετάβασης και εκπομπής για το στοιχισμένο dataset. Σε πρώτη φάση, δημιουργεί και αρχικοποιεί μηδενικούς πίνακες transition_counts και emission_counts. Για κάθε ακολουθία του συνόλου A, ενημερώνει τους πίνακες μετρήσεων ανάλογα με το αν το σύμβολο είναι _ (Delete) ή όχι (Match). Μόλις το κάνει αυτό, μετατρέπει τις μετρήσεις σε πιθανότητες διαιρώντας με το άθροισμα των γραμμών κάθε πίνακα. Επιστρέφει τις πιθανότητες μετάβασης και εκπομπής στον constructor της κλάσης.

Ακόμα, έχουμε τη μέθοδο calculate_hmm_sequence(), η οποία καλείται απευθείας από την main του προγράμματος, και υπολογίζει και επιστρέφει την ακολουθία των καταστάσεων του Hidden Markov Model Profile. Το κάνει αυτό, ελέγχοντας, για κάθε στήλη στις στοιχισμένες ακολουθίες, αν υπάρχουν διαγραφές, δηλαδή εμφάνιση του συμβόλου «_» ή όχι, και βάση αυτού καθορίζοντας την κατάσταση την οποία θα εισάγει κάθε φορά στην αντίστοιχη θέση του HMM profile (Match – M, Delete – D, ή και τις δύο).

Επιπρόσθετα, έχουμε την μέθοδο align_dataset, η οποία καλείται επίσης από την main, και η οποία υπολογίζει και επιστρέφει τις βαθμολογίες στοίχισης και τις διαδρομές στοίχισης, για τις συμβολοσειρές του συνόλου B. Η μέθοδος αυτή

είναι ιδιαίτερα σύντομη καθώς απλά καλεί την μέθοδο forward_algorithm() για την εύρεση των διαδρομών στοίχισης και την μέθοδο viterbi_algorithm() για την εύρεση των διαδρομών στοίχισης. Και οι δύο αυτές μέθοδοι θα αναλυθούν εν συνεχεία. Εκείνη, επιστρέφει τα αποτελέσματα στη main, χωρίς να τα επεξεργαστεί περαιτέρω.

Η επόμενη μέθοδος μας είναι η forward_algorithm(), η οποία υπολογίζει τις βαθμολογίες στοίχισης για το δοσμένο σύνολο συμβολοσειρών (εδώ πρόκειται για το σύνολο B), με βάση τις πιθανότητες εκπομπής και μετάβασης. Εν αρχή, δημιουργεί και αρχικοποιεί έναν πίνακα πιθανοτήτων, τον fwd. Έπειτα, υπολογίζει τις πιθανότητες για κάθε θέση στην ακολουθία, χρησιμοποιώντας τις πιθανότητες μετάβασης και εκπομπής, οι οποίες βρίσκονται αποθηκευμένες στα αντίστοιχα attributes της κλάσης, γεμίζοντας, με αυτόν τον τρόπο τον παραπάνω πίνακα. Στο τέλος, επιστρέφει στη μέθοδο align_dataset() το άθροισμα των πιθανοτήτων για κάθε ακολουθία.

Η τελική μέθοδος της κλάσης και ολόκληρου του προγράμματος είναι η viterbi_algorithm(), η οποία υπολογίζει τις διαδρομές στοίχισης για το δοσμένο σύνολο συμβολοσειρών (εδώ πρόκειται για το σύνολο B), με βάση τις πιθανότητες εκπομπής και μετάβασης. Σε πρώτο στάδιο, δημιουργεί και αρχικοποιεί έναν πίνακα πιθανοτήτων, τον viterbi, καθώς και έναν πίνακα δεικτών επιστροφής, τον backpointer. Συνεχίζει, υπολογίζοντας τις πιθανότητες για κάθε θέση στην ακολουθία, κρατώντας πάντα την καλύτερη διαδρομή. Στο τέλος της, επιστρέφει στη μέθοδο align_dataset() τις διαδρομές στοίχισης για κάθε ακολουθία.

Συμπεράσματα

Συνολικά, μπορούμε να πούμε πως υλοποιήσαμε όλα τα ζητούμενα της εργασίας σε πολύ ικανοποιητικό βαθμό. Διαπιστώσαμε πως η τεχνικές του Multiple Sequence Alignment και του HMM Profiling, μπορούν να μας βοηθήσουν να παρατηρήσουμε, με επιστημονικό και τεκμηριωμένο τρόπο, διάφορες συσχετίσεις και ομοιότητες μεταξύ δεδομένων, τα οποία με μια επιφανειακή ματιά μπορεί να έμοιαζαν ετερογενή και ανομοιόμορφα. Ακόμα, η εργασία αυτή μας έδωσε βοήθησε να κατανοήσουμε καλύτερα τη σημασία των εννοιών που αναλύσαμε στο μάθημα, καθώς ασχοληθήκαμε με πρακτικές τους εφαρμογές που συνδέονται άμεσα με την επιστήμη της Βιοπληροφορικής.

Μέσα στον πηγαίο κώδικα, έχουμε παραθέσει χρήσιμα σχόλια ανά τακτά χρονικά διαστήματα, όπου αυτά κρίθηκαν απαραίτητα, ώστε το πρόγραμμα να είναι πιο ευανάγνωστο και κατανοητό για τον αναγνώστη.

Δομή παραδοτέου:

Αρχεία πηγαίου κώδικα (τα αρχεία .idea, __pycache__, και venv μπορούν να αγνοηθούν, καθώς αποτελούν αρχεία για τη λειτουργία του project από το IDE που χρησιμοποιήθηκε (PyCharm) και όχι αρχεία στο πηγαίου κώδικα):

source2024.zip → main.py (αρχείο εκτέλεσης του προγράμματος)

source2024.zip → generation.py

source2024.zip → alignment.py

και:

source2024.zip → hmm.py

Συμπληρωματικά αρχεία (screenshot από την εκτέλεση του προγράμματος στο τερματικό):

auxiliary2024.zip → program_execution_screenshot_1.png

auxiliary2024.zip → program_execution_screenshot_2.png

auxiliary2024.zip → program_execution_screenshot_3.png

και:

auxiliary2024.zip → program_execution_screenshot_4.png

Σε περίπτωση που προκύψει οποιοδήποτε πρόβλημα, παρακαλούμε επικοινωνήστε μαζί μας.