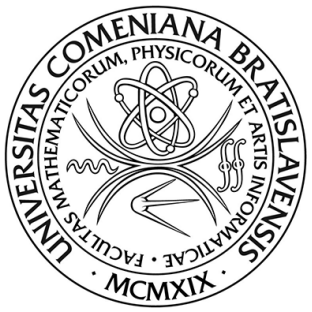
Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Trénovanie parametrov pre hľadanie génov s využitím externej informácie

Diplomová práca

Bratislava 2014 Bc. Lukáš Bača

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky



Trénovanie parametrov pre hľadanie génov s využitím externej informácie

Diplomová práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	9.2.9. aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	Mgr. Broňa Brejová PhD.

Bratislava 2014

Bc. Lukáš Bača

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Lukáš Bača
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: 9.2.9. aplikovaná informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský

Názov: Trénovanie parametrov pre hľadanie génov s využitím externej informácie

Cieľ: Jednou zo základných úloh spracovania DNA sekvencií je identifikácia génov kódujúcich proteíny. Na tento problém sa používajú pravdepodobnostné modely, ktorých parametre je potrebné prispôsobiť pre rôzne skupiny organizmov. Cieľom diplomovej práce je navrhnúť a porovnať metódy na výber trénovacej množiny pre určovanie potrebných parametrov. Táto trénovacia množina sa vyberá z automaticky predpovedaných génov na základe ich porovnania s dostupnými experimentálnymi dátami.

Vedúci: Mgr. Bronislava Brejová, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. PhDr. Ján Rybár, PhD.

Dátum zadania: 13.10.2010

Dátum schválenia: 09.11.2010

doc. Ing. Igor Farkaš, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Čestné vyhlásenie

Prehlasujem, že som svoju diplomovú prácu vypracoval samostatne s použitím uvedených zdrojov.

.....

Bc. Lukáš Bača

Poďakovanie

Poďakovanie patrí najmä mojej vedúcej diplomovej práce Mgr. Broni Brejovej PhD. za vedenie, cenné rady a pripomienky i optimistický prístup, ako aj mojej rodine a priateľom, ktorí ma podporovali v celom mojom štúdiu.

Abstrakt

Bača, Lukáš: Trénovanie parametrov pre hľadanie génov s využitím externej informácie [Diplomová práca]. Univerzita Komenského v Bratislave. Fakulta matematiky, fyziky a informatiky; Katedra aplikovanej informatiky. Vedúci diplomovej práce: Mgr. Broňa Brejová PhD. Bratislava: FMFI UK, 2014.

V našej diplomovej práci sme sa zamerali na návrh nového spôsobu výberu trénovacej množiny pre skrytý Markovov model určený na hľadanie génov. Rozhodli sme sa naviazať na prácu [3, Brejova et al.] a za cieľ sme si stanovili zvýšenie presnosti predpovede génov novo osekvenovaných genómoch za pomoci externých informácií. Navrhli

sme nový spôsob výberu trénovacej množiny pre skrytý Markovov model a na zlepšenie predikcie sme použili externé informácie ako EST a proteíny. Problém sme riešili natrénovaním klasifikátora SVM na vopred oanotovanom druhu D.melanogaster. Podarilo sa nám ukázať, že nami vytvorená metóda vie vyfiltrovať kvalitnú trénovaciu množinu, ktorá vedie k podstatnému zlepšeniu HMM modelu pre program Snap. V budúcnosti by sme sa chceli venovať zväčšeniu množstva čŕt, čo môže zlepšiť výsledky trénovaného SVM modelu.

Kľúčové slová: gén, anotácia DNA, skrytý Markovov model, ExonHunter, Maker, externá informácia, klasifikácia

Abstract

Bača, Lukáš: Train parameters for finding genes using external information [Diploma thesis].
Comenius University in Bratislava. Faculty of Mathematics, Physics and Informatics; Department of Applied Informatics . Diploma thesis supervisor: Mgr. Broňa Brejová PhD. Bratislava: FMFI UK, 2014.

In our thesis we focused on the proposal for a new training set selection method for hidden Markov models used in gene finding.
We decided to build on the work of [3, Brejova et al.] and we have aimed to set increase the accuracy of gene predictions newly sequenced genomes using external information. We have proposed a new method of training set selection for hidden Markov models and to improve prediction, we used external information such as EST and proteins. The problem was solved training SVM classifier to pre-annotated genomes D.melanogaster. We managed to show that our developed method can filter out high-quality training set, which leads to a significant improvement hmm model for snap. In the future we would like to pay increased amounts on the basis of features, which are trained SVM.

Keywords: gene, DNA anotation, Hidden Markov model, ExonHunter, Maker, external evidence, clasification

Obsah

Zoznam obrázkov

Zoznam tabuliek

Kapitola 1

Úvod

Hľadanie génov kódujících proteíny v DNA sekvenciách je základným bioinformatickým problémom. Na riešenie tohoto problému sa používajú pravdepodobnostné modely. Každý z týchto modelov obsahuje veľké množstvo parametrov, ktoré je potrebné nastaviť pre každý novoosekvenovaný organizmus zvlášť. V tejto diplomovej práci sa budeme zaoberať návrhom a porovnávaním jednotlivých metód pre výber trénovacej množiny pre určenie týchto parametrov. Na výber tejto množiny budeme používať experimentálne dáta.

V našej práci si na úvod ozrejmíme základné biologické pojmy, ktoré budeme používať v ostatných častiach. Následne uvedieme problém hľadania génov a ukážeme modely, za pomoci ktorých tento problém môžeme riešiť. Pokračovať budeme vysvetlením, čo sú to externé informácie a ako nám pomáhajú pri zvyšovaní presnosti predpovedí génov. Na konci kapitoly spomenieme trénovanie takýchto modelov.

V ďalšej kapitole popisujeme výsledky jednoduchších experimentov s použitím externých informácií, pričom v každom experimente používame len jeden typ externej informácie.

V poslednej kapitole predstavíme systém, ktorý dokáže spojiť viac typov informácií metódami strojového učenia, konkrétne support vector machine(SVM)

Kapitola 2

Hľadanie génov

V tejto kapitole popisujeme úvod do problematiky hľadania génov, počnúc základnými pojmami, cez prístupy využívajúce skryté Markovove modely až po trénovanie modelov s využitím externých informácií.

2.1 Základné pojmy

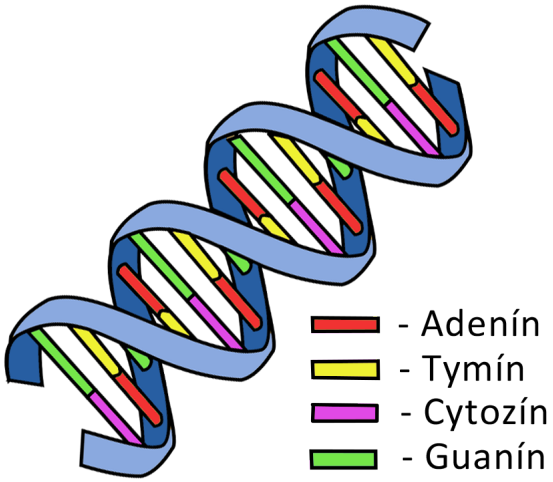
DNA (deoxyribonukleová kyselina) je prírodný polymér, v ktorom sú ukladané genetické informácie v živých organizmoch. V DNA sú zapísané predpisy na tvorbu proteínov, funkčných RNA molekúl, ale aj informácie, ktoré riadia kedy a koľko proteínov sa má tvoriť v bunke. V prípade eukaryotických organizmov sa DNA nachádza v bunkovom jadre, zatiaľ čo u prokaryotických organizmov sa nachádza v cytoplazme. DNA väčšinou nadobúda tvar dvojzávitnicovej špirály ako na obrázku 2.1.

Chromozóm je súvislý úsek DNA. V ľudskom organizme sa nachádza 22 párov chromozómov a dva pohlavné.

Nukleotid je základnou stavebnou jednotkou DNA a nositeľom informácie uloženej v nej. Táto informácia je uložená v nukleovej báze. Jednotlivé nukleotidy sa odlišujú navzájom len v nukleovej báze.

Poznáme 4 základné nukleové bázy: Adenín, Cytozín, Guanín, Tymín. DNA teda reprezentujeme ako reťazec nad štvorpísmeovou abecedou, ktorú dostaneme zo začiatočných písmen štyroch nukleových báz

$$\sum = \{A, C, T, G\}$$



Obr. 2.1: DNA

Poradie báz v DNA reťazci tvorí genetickú informáciu. Dve vlákna DNA sú k sebe antiparalelné a sú spojené vodíkovými väzbami. Tieto vodíkové väzby sa vytvárajú medzi dvojicami báz, ktoré nazývame komplementárne:

- Guanín-Cytozín: viažu sa tromi vodíkovými väzbami
- Adenín-Tymín: viažu sa dvomi vodíkovými väzbami

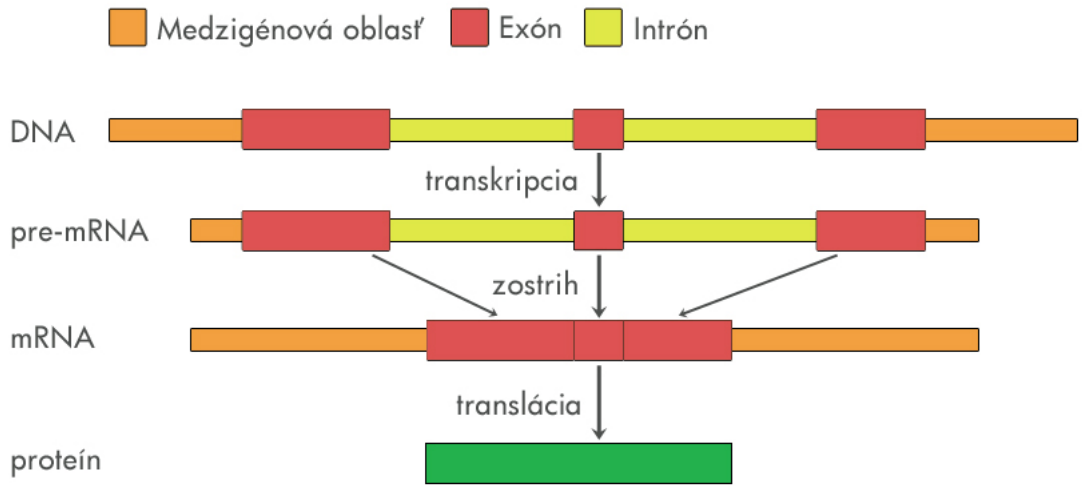
Jednotlivé konce DNA reťazca nazývame 5' koniec a 3' koniec. Smer čítania reťazca DNA od 5' do 3' nazývame kladný smer a opačný smer nazývame záporný. Proces, pri ktorom získavame DNA sekvenciu z organizmu, nazývame sekvenovanie. Toto získavanie ale nie je priame. Dĺžka ľudského genómu je 3.2 giga báz, pričom nevieme na jedenkrát prečítať celú dĺžku, ale len úseky o dĺžke 300-800 báz.

Proteíny sú základom všetkých známych organizmov a participujú na každom procese v bunke. Slúžia ako receptory vonkajších chemických alebo elektrických signálov, na prenos nevyhnutných látok pre zachovanie života v organizme, ako biokatalyzátory chemických reakcií. Na stavbe proteínov sa podieľa 20 aminokyselín. Rovnako ako pri bázach je každej aminokyeline priradené jedno písmeno z abecedy:

$$\sum = \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$$

Pri proteínoch je dôležité nielen to, z akých aminokyselín sa skladajú a ich poradie, ale aj ich trojrozmerná štruktúra v priestore.

Gén je časť DNA sekvencie slúžiaca na vytvorenie proteínu (Obrázok 2.2).
Gén pozostáva zo striedajúcich sa úsekov nazývaných exóny a intróny. Exóny sú úseky, ktoré kódujú proteín a intróny sú časti, ktoré sa pred prekladom vystrihujú.



Obr. 2.2: Genetický kód

Každá aminokyselina proteínu je kódovaná jedným kodónom, ktorý tvoria 3 bázy. Pomocou 3 nukleotidov vieme vytvoriť kombináciu 64 kodónov. Ale aminokyselín, ktore kódujú, je len 20. Z toho vyplýva, že niektoré aminokyseliny sú zakódované aj viacerými rôznymi kodónmi (tabuľka 2.1). V prípade, že nastane mutácia v kodóne, ktorá nemá za následok zmenu aminokyseliny, výsledný proteín bude mať rovnakú funkciu.

Expresia génov kódujúcich proteíny je preklad DNA na proteín pozostávajúci z naskedujúcich krokov:

Ala	A	GCU, GCC, GCA, GCG	Lys	K	AAA, AAG
Arg	R	CGU, CGC, CGA, CGG, AGA, AGG	Phe	F	UUU, UUC
Ser	S	UCU, UCC, UCA, UCG, AGU, AGC	Asn	N	AAU, AAC
Asp	D	GAU, GAC	Pro	P	CCU, CCC, CCA, CCG
Gln	Q	CAA, CAG	Cys	C	UGU, UGC
Glu	E	GAA, GAG	Met	M	AUG
Gly	G	GGU, GGC, GGA, GGG	Thr	T	ACU ACC, ACA, ACG
His	H	CAU, CAC	Trp	W	UGG
Ile	I	AUU, AUC, AUA	Tyr	Y	UAU, UAC
Leu	L	CUU, CUC, CUA, CUG, UUA, UUG	Val	V	GUU, GUC, GUA, GUG
Stop		UAA, UAG, UGA			

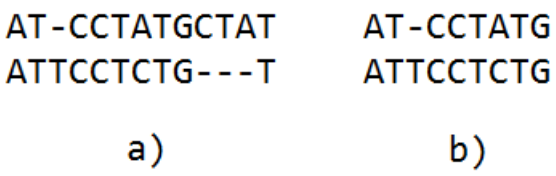
Tabuľka 2.1: Genetický kód

- Transkripcia je skopírovanie súvislého úseku DNA pričom vznikne pre-mRNA (precursor mRNA). Pri tomto kroku sa zamieňa báza Tymín za bázu Uracil.
- Zostrih je vystrihnutie intrónov z pre-mRNA. Jeho výsledkom je mRNA (messenger RNA).
- Translácia je preklad mRNA na aminokyseliny. Tento preklad sa vykonáva po trojiciach (kodónoch)

Na začiatku a konci každého génu sa nachádza štart a stop kodón. Štart kodón slúži ako signál pre začatie prekladu a stop kodón na koniec prekladu. Tieto časti nám slúžia pri translácii na určenie okrajov. Na rozlíšenie uzavretých úsekov intrónov a exónov v géne nám súžia signály donor a akceptor. Donor je signál, ktorý určuje začiatok intrónu a je zvyčajne tvorený nukleotidmi GT. Akceptor určuje zas koniec intrónu a je tvorený dvojicou nukleotidov AG. GT a AT sa však môžu vyskytnúť aj v iných miestach génu. Tieto signály sa využívajú pri zostrihu.

Zarovnanie je metóda v bioinformatike, ktorá slúži na nájdenie podobností medzi dvoma sekvenciami, t.j. snažíme sa spárovať nukleotidy z jednej sekvencie s nukleotidmi z druhej sekvencie. Pri tomto procese môžu nastať situácie:

- **zhoda** - je miesto, na ktorom sa v oboch sekvenciách nachádza rovnaký nukleotid.
- **mutácia** - zodpovedá spárovanému miestu, na ktorom sa nachádzajú rôzne nukleotidy.
- **inzercia** - je miesto, na ktorom je nukleotid v jednej sekvencii spárovaný s medzerou v druhej sekvencii.
- **delécia** - je opak inzercie, teda spárovaná je medzera s nukleotidom.



Obr. 2.3: Jednoduché zarovnanie sekvencií X=ATCCTATGCTAT a Y=ATTCCTCTGT: a) globálne b) lokálne

Na ohodnotenie, aké dobré dané zarovnanie je, používame skórovacie schémy. Jednou z najednoduchších je

- zhoda +1
- nezhoda -1
- medzera -1

Pri zarovnaní uvažujeme dve vstupné sekvencie $X = x_1, ..., x_m$ a $Y = y_1, ..., y_n$. Poznáme dva základné druhy zarovnaní:

1. **lokálne**, v ktorom sa snažíme nájsť zarovnanie podsekvencie $x_i, ..., x_j$ z X s podsekvenciou $y_k, ..., y_l$ z Y, ktoré majú najväčšie skóre. Napríklad na obrázku 2.3/b vidíme zarovnanie zo skóre 5, ak použijeme predošlú jednoduchú skórovaciu metódu.
2. **globálne**, v ktorom sa snažíme nájsť zarovnanie pre celé sekvencie X a Y, ktoré bude mať maximálne skóre. Na obrázku 2.3/a je zarovnanie zo skóre 3, v prípade, že použijeme predošlú jednoduchú skórovaciu metódu.

Na riešenie problému globálneho zarovnania môžeme použiť dynamické programovanie. Vytvoríme si dvojrozmernú tabuľku A, kde počet riadkov zodpovedá dĺžke sekvencie X plus jedna a počet stĺpcov dĺžke sekvencie Y plus jedna. Nulté riadky tabuľky nastavíme podľa vzťahu:

$$A[0, j] = -j; A[i, 0] = -i$$

Počas vypĺňania tabuľky nám môžu nastať štyri prípady:

$x_i = y_j$ sekvencia X na pozícii x_i a sekvencia Y na pozícii y_j sa rovnajú, potom $A[i, j] = A[i - 1, j - 1] + 1$.

$x_i \neq y_j$ sekvencia X na pozícii x_i a sekvencia Y na pozícii y_j sa nerovnajú, potom $A[i,j] = A[i-1,j-1] - 1$.

Sekvencia X na pozícii x_i je zarovnaná s medzerou, potom $A[i,j] = A[i-1,j] - 1$.

Sekvencia Y na pozícii y_j je zarovnaná s medzerou potom $A[i,j] = A[i,j-1] - 1$.

Pre každý prvok tabuľky A vypočítame maximum zo všetých predošlých podmienok. V každom prvku tabuľky si zapamätáme ešte informáciu, z ktorého prvku bola vypočítaná (zľava, zhora, diagonálne). Globálne zarovnanie je cesta z pravého dolného rohu dolavého horného rohu po týchto zapamätaných spätných šípkach.

V prípade, že chceme počítať lokálne zarovnanie, stačí pridať piaty prípad:
$A[i,j] = 0$ zodpovedajúci prázdnemu zarovnaniu.

Lokálne zarovnanie nájdeme ako cestu od najväčšieho prvku (prípadne od všetkých prvkov väčších ako S) po najbližšiu 0.

Časová a priestorová zložitosť obidvoch algoritmov je $O(|X| * |Y|)$, čo vyplýva z veľkosti tabuľky a konštantného výpočtu jedného prvku tabuľky.

2.2 Problém hľadania génov

Hľadanie génov je jedným z hlavných bioinformatických problémov. Jeho cieľom je nájdenie všetkých génov skúmaného organizmu. To znamená v sekvencii DNA každý nukleotid priradiť do jednej z troch kategórií (Obrázok 2.2):

- Medzigénová oblasť
- Intrón
- Exón

Nájdené gény môžu biológovia dalej skúmať a určovať ich funkcie v bunke.

Aby sme vedeli vyhodnotiť, ako je daná metóda na hľadanie génov úspešná, porovnáme predpovedané gény so skutočnými na dobre anotovanej sekvencii a použijeme nasledovné metriky:

- **Špecifickosť** vyjadruje, koľko percent z nájdených položiek je v skutočnosti správnych. Číslo 100% by vyjadrovalo, že všetky nami označené nálezy sú v skutočnosti správne.

$$\check{s}pecifickos\check{t} = \frac{|spr\acute{a}vne\ predikovan\acute{e}|}{|v\check{s}etky\ predikovan\acute{e}|}$$

- **Senzitivita** vyjadruje, koľko percent zo všetkých skutočných položiek sme našli.

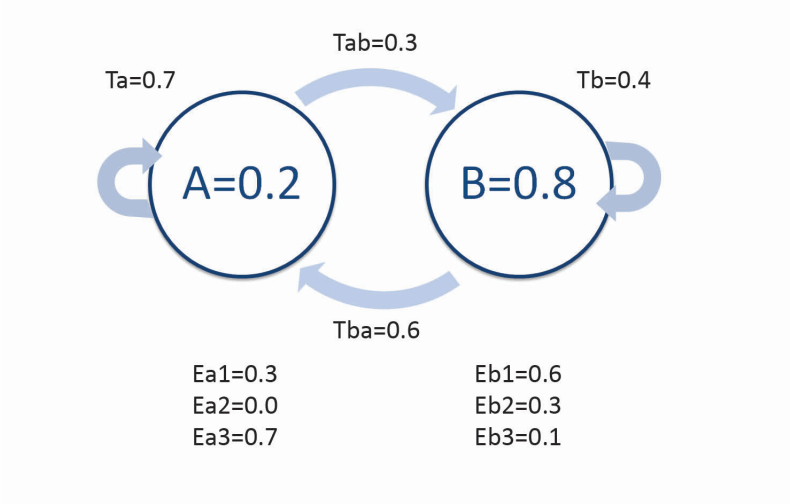
$$\check{s}pecificita = \frac{|spr\acute{a}vne\ predikovan\acute{e}|}{|v\check{s}etky\ skuto\check{c}n\acute{e}|}$$

Pri navrhovaní metódy sa snažíme, aby boli obidve metriky čo najvyššie. To znamená, že na jednej strane chceme nájsť čo najviac skutočných položiek, súčasne sa však snažíme dosiahnuť, aby všetky nami označené nálezy boli označené správne. Ako položky môžme uvažovať napríklad celé gény, exóny alebo jednotlivé nukleotidy v exónoch.

2.3 Skryté Markovove modely

Skryté Markovove modely (HMM z ang. Hidden Markovov Model) sú štatistické modely stochastických procesov na počítanie pravdepodobností na základe pozorovaní. Základné vlastnosti týchto modelov sú:

- **konečný počet stavov** - tieto stavy nie sú navonok pozorovateľné; počas behu modelu nevie externý pozorovateľ zistiť, v ktorom z týchto stavov sa model nachádza. Na obrázku 2.4 sú stavy A a B.
- **prechody medzi stavmi** - sú znázornené ako ohodnotené a orientované hrany medzi jednotlivými stavmi. Orientácia určuje smer prechodu a ohodnotenie určuje, s akou pravdepodobnosťou prejde model do stavu určeného orientáciou. Súčet hodnotení hrán vychádzajúci z jedného stavu musí byť 1 (100%). Na obrázku 2.4 sú to ohodnotenia Tab, Tba, Ta, Tb.
- **emisné pravdepodobnosti** - každý skrytý stav navonok v jednom kroku emituje jeden z možných symbolov s pravdepodobnosťou podľa tabuľky. Tento symbol je pre pozorovateľa viditeľný. HMM na obrázku 2.4 emituje tri symboly (1, 2, 3), kde Ea1 je emisná pravdepodobnosť emitovania symbolu 1 skrytým stavom A.
- **počiatočná pravdepodobnosť stavov** - určuje, v ktorom stave HMM začína alebo končí. Na obrázku 2.4 v stave A začína s pravdepodobnosťou 20% a v stave B s pravdepodobnosťou 80%.



Obr. 2.4: Príklad HMM s dvoma stavmi

V HMM vieme riešiť 2 základné úlohy, ktoré popíšeme nižšie:

Nájdenie najpravdepodobnejšej postupnosti stavov podľa danej postupnosti pozorovaní

Nájdenie najpravdepodobnejšej postupnosti stavov $S_1 ... S_n$ v prípade daných pozorovaní X $x_1, ..., x_n$ môžeme vyjadriť ako $\arg\max_S Pr(S, X)$. Pravdepodobnosť vygenerovania postupnosti X a súčasné prejdenie stavov S vypočítame ako:

$$Pr(S, X) = t_{S_1} e_{S_1,X_1} \prod_{i=2}^{n} t_{S_{i-1},S_i} e_{S_i,X_i},$$

kde člen t_{S_1} predstavuje počiatočnú pravdepodobnosť stavu s_1 a e_{S_1,X_1} pozorovanie v počiatočnom stave. Následne pre všetkých ostatných n stavov t_{S_{i-1},S_i} pravdepodobnosť prechodu z minulého stavu do aktuálneho a e_{S_i,X_i} pravdepodobnosti pozorovania v tomto stave.

Skryté Markovove modely sú najčastejšie používanou metódou na hľadanie génov a budeme ich využívať aj tejto diplomovej práci.

Vypočítanie najpravdepodobnejšej postupnosti stavov pomocou "hrubej sily"by zabralo veľmi dlhý čas. Preto využijeme dynamické programovanie, konkrétne Viterbiho algoritmus. Vytvoríme si dvojrozmernú tabuľku, kde počet riadkov zodpovedá dĺžke vstupnej sekvencie a stĺpce stavom, ktoré sa nachádzajú v našom modeli. Takto vzniknú členy $A_{i,j}$.

Prvý riadok vypočítame ako $A_{1,j} = t_j e_j, x_1$

Ostatné riadky spočítame vzorcom $A_{i,j} = max_k A_{i-1,k}\ t_{k,j}\ e_j, x_i$

Hľadáme maximum, ktoré vypočítame zo stĺpcov v predchádzajúcom riadku vynásobených prechodom z minulého stavu do aktuálneho stavu a emisiou v aktuálnom stave.

Priestorová zložitosť algoritmu zodpovedá tabuľke, ktorú si musíme vyplniť. Tabulka má veľkosť počet stavov(|A|) krát dĺžka vstupnej sekvencie(n). Zložitosť je teda $O(|A| * n)$. Časová zložitosť tohto algoritmu je $O(|A|^2 * n)$.

Trénovanie HMM

Ako vstup máme k trénovacích postupností stavov a sekvencií, ktoré vygenerovali $(S_1, X_1), (S_2, X_2), ..., (S_k, X_k)$

Našou úlohou je nájsť parametre modelu, počiatočnú pravdepodobnosť t_i, prechodové pravdepodobnosti $t_{i,j}$ a emisné pravdepodobnosti $e_{i,x}$. Pritom chceme, aby súčin pravdepodobností vygenerovania daných trénovacích sekvencií bol čo najväčší. Aby sme dostali emisné pravdepodobnosti stačí spočítať frekvencie výskytov znaku x emitovaného zo stavu i.

$$e_{i,x} = \frac{E_{i,x}}{\sum_b E_{k,b}},$$

kde $E_{i,x}$ predstavuje, koľkokrát je emitovaný znak x v stave i a $\sum_{b_i} E_{k,b_i}$ počet emisií v stave i. V prípade, že chceme dostať prechodové pravdepodobnosti, stačí spočítať frekvencie prechodov zo stavu i do stavu j.

$$T_{i,j} = \frac{A_{i,j}}{\sum_j A_{i,j}},$$

kde $T_{i,j}$ predstavuje počet prechodov zo stavu i do j a $\sum_j A_{i,j}$ počet všetkých prechodových pravdepodobností, ktoré vychádzajú zo stavu i.

Problém môže nastať v prípade, že niektorý z prechodov alebo emisií nemá žiadny alebo len veľmi malý počet trénovacích príkladov. V prípade žiadnych dát vznikne nevypočítateľný zlomok $(\frac{0}{0})$ a v prípade malého počtu bude pravdepodobnosť oproti ostatným podhodnotená.

V prípade, že nemáme trénovacie príklady ako v predošlom prípade, ale len postupnosť vygenerovaných symbolov, môžme na trénovanie použiť Baum-Welchov algoritmus alebo Viterbiho trénovanie.

Baum-Welchov Algoritmus[16] - jeho cieľom je nájdenie parametrov HMM modelu takých, že maximalizujú pravdepodobnosť vygenerovania sekvencie danej postupnosti

vygenerovaných symbolov, pričom nepoznáme postupnosť stavov. Je to iteratívny algoritmus bez učiteľa. To v našom prípade znamená, že na vstupe máme sekvencie DNA a jej predpokladané anotácie, ktoré nemusia byť pre každý prvok správne.

Pred prvým behom algoritmu musíme nastaviť počiatočné váhy v HMM modeli (emisné a prechodové pravdepodobnosti) $\Theta^{(0)}$. Toto nastavenie je možné dvomi spôsobmi, a to buď na náhodné čísla alebo na čísla, ktoré sú pre model predpokladané. Prvá možnosť je dobrá v prípade, že nemáme o modeli dodatočné informácie. Druhá možnosť môže urýchliť beh algoritmu tým, že vzdialenosťi, ku ktorým bude konvergovať bude menšia.

Algoritmus je rozdelený na dve časti, ktoré sa opakujú, až kým nenastane konvergencia do lokálneho minima:

- **E-krok** je dopredný krok, pri ktorom vypočítame pre každú trénovaciu postupnosť:

  $Q_i^{(k)}(S^{(i)}) = Pr(S^{(i)} = s^{(i)}|X^{(i)} = s^{(i)}, \Theta^{(k)}),$

  kde k zodpovedá $k-$temu behu programu, i zodpovedá $i-$temu trénovaciemu setu a Q zodpovedá očakávanej hodnote pravdepodobnostnej funkcie. Na výpočet použijeme dopredný algoritmus, ktorý sa podobá na Viterbiho algoritmus.

- **M-krok** je spätný krok, kedy sa snažíme vypočítať nové váhy Θ, ktoré budú zväčšovať hodnotu pravdepodobností vygenerovaných dát.

  $\Theta^{k+1} = \sum_{i=1}^{t} \sum_{s(i)} [Q_i^{(k)}(s^{(i)}) * Pr(X^{(i)} = x^{(i)}, S^{(i)} = s^{(i)}|\Theta^k)]$

Tieto kroky opakujeme až do momentu, kým vierohodnosť váh Θ^k je väčšia ako Θ^{k-1}.

Vierbiho trénovanie[16] je druhý možný algoritmus na nájdenie parametrou HMM. Rozdiel oproti B-W je v tom, že tento algoritmus sa snaží počítať iba malý počet najpravdepodobnejších ciest oproti všetkým možným v algoritme Baum-Welch.
V prvom kroku pre každú vstupnú sekvenciu nájdeme postupnosť stavov za pomoci viterbiho algoritmu.
Zadefinujeme si pravdepodobnostnú mieru P^V :

$$P^V(X|\Theta) = \frac{P(S*(X)|\Theta)}{\sum_O P(S*(X)|\Theta)},$$

kde $P(S*(X)|\Theta)$ je pravdepodobnosť najlepšej cesty zo sekvencie X na základe parametrov Θ.
A $\sum_X P(S*(X)|\Theta)$ je súčet pravdepodobností všetkých vstupných sekvencií

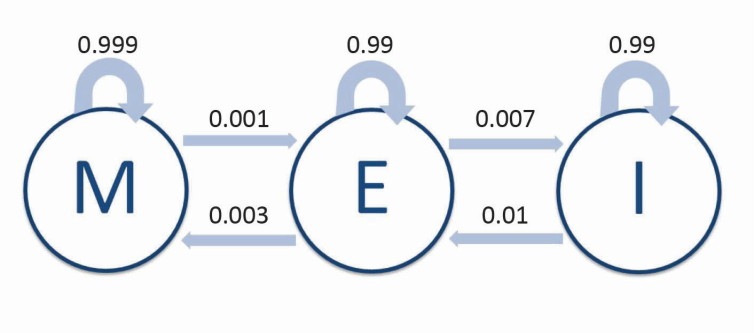
A viterbiho trénovanie sa snaží minimalizovať:

$E = \sum_{k=1}^{K} -\log P^V(X_k|\Theta)$

Je suma záporných logaritmov pravdepodobnostných mier pre všetky vstupné sekvencie. E je nespojité lebo parametre Θ (prechodové a emisné pravdepodobnosti) sa počas behu menia a $S*$ sa môže menit skokovo.[15]

2.4 Hľadanie génov pomocov HMM

Na hľadanie génov sa najčastejšie využíva HMM. Jednoduché HMM na hľadanie génov by mohlo mať tri stavy, zodpovedajúce medzigénovým častiam (M), exónom (E) a intrónom (I) (obrázok 2.5). Každý stav emituje 4 základné nukleotidy DNA $\{A, C, T, G\}$



Obr. 2.5: Príklad jednoduchého HMM na hľadanie génov

Prechod medzi intrónom a medzigénovou časťou je zakázaný, lebo takéto oblasti nemôžu v DNA ísť za sebou.

V skutočnosti sa ale využívajú oveľa zložitejšie HMM modely podobne ako na obrázku 2.6. Toto HMM má dve skoro identické časti, založené na smere prepisu (transkripcie) po vlákne DNA:

- **forward strand** je priame vlákno. Smer jeho prekladu je od 5' konca ku 3' koncu. 5' a 3' predstavujú označenia koncov prekladaných úsekov. Ich názov je

odvodený od uhlíkových pozícií.Takýto gén začína start kodónom a končí stop kodónom.

- **reverse strand** je komplementárne vlákno k priamemu. Aj keď v skutočnosti translácia prebieha na komplementárnom vlákne od 5' do 3', z pohľadu priameho vlákna ide o smer od 3' do 5'. V prípade, že priame vlákno prezeráme v smere od 5' do 3', nevedeli by sme nájsť takýto gén z dôvodu, že bude začínať stop kodónom, končiť start kodónom a umiestnenie donor a akceptor signálov bude tiež v opačnom poradí. Navyše všetky bázy budú komplementárne, čo znamená, že start kodón ATG bude v skutočnosti CAT.

Takýto model nám teda umožnuje hľadať gény v priamom aj reverznom smere bez nutnosti púšťať program na hľadanie druhý raz na opačnom vlákne.

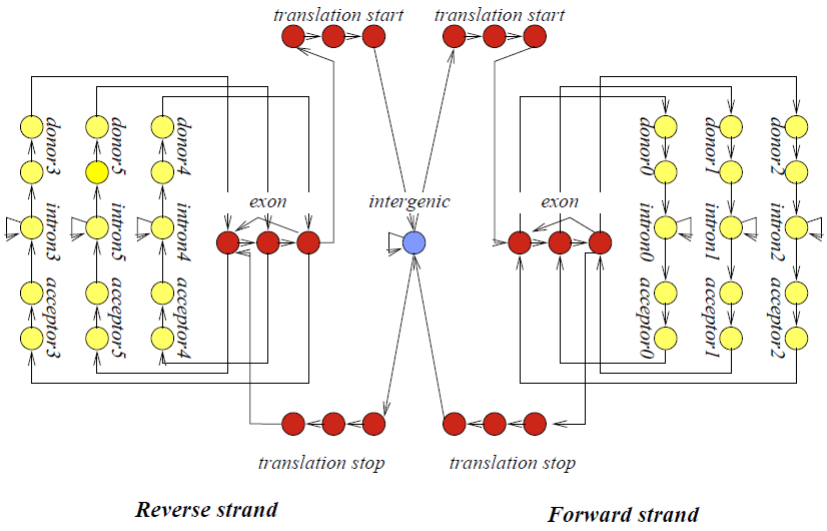
Ďalším vylepšením je vytvorenie trojstavových podmodelov pre exón, ktoré reprezentujú kodón. Vďaka tomu vieme priradiť rôzne emisné pravdepodobnosti pre rôzne časti kodónu.

Aby bolo zachované poradie, v ktorom sa striedajú kodóny u exóne aj po tom, ako sa exónová časť preruší intrónom, stavy pre intróny sú tiež v troch kópiach, z ktorých každá zodpovedá intrónu po určitej pozícii v kodóne. Po skončeni intrónu sa napája do nasledujúceho exónu, ktorý má pokračovať v kodóne ako keby nebol prerušený intrónom. Toto jednoduché pravidlo spôsobí, že budú zachované trojprvkové kodóny aj v prípade, že sú v strede prerušené intrónom.

Na presnejšie určenie prítomnosti intrónu model obsahuje aj stavy pre signály donor a akceptor. Toto pomáha lepšie nastaviť emisné pravdepodobnosti pre jednotlivé časti génu a tak spresniť model.

Ďalšou možnosťou pri HMM sú stavy vyšších rádov. Toto si vieme predstaviť ako okno do minulosti, ktoré si pamätáme pri vyhodnocovaní aktuálneho stavu reprezentované FIFO (z angl. first in first out) zásobníkom obsahujúcim posledných k vygenerovaných stavov. Pri prejdení na ďalší stav zabúdame posledne pamätaný symbol a na prvé miesto uložíme vygenerovaný symbol.

- **Nultý rád** je totožný s doteraz spomínaným HMM. Emisiu určuje pravdepodobnosť $\Pr(X_i|S_i)$. Veľkosť tabuľky emisií je |S| * |A|, kde |S| predstavuje počet emitovaných znakov(v prípade |S|=4) a |A| predstavuje počet stavov v modeli.



Obr. 2.6: Príklad HMM na hľadanie génov [3]

- **n. rád** predstavuje počet symbolov, ktoré si pamätáme do minulosti. Emisiu určuje pravdepodobnosť $\Pr(s_i|x_i, s_{i-1}, ..., s_{i-n})$. Veľkosť tabuľky emisií narastie na $|S|^{n+1} * |A|$

Ďalšou možnosťou pri HMM je použitie **HSMM** (z ang. Hidden semi-Markov model; skrytý polo-Markov model). V tomto modeli je možné v jednom kroku vygenerovať viaceré symboly súčasne v jednom stave. Zjednodušene si to môžeme predstaviť, že najskôr sa vygeneruje počet krokov k, počas ktorých zostane model v jednom stave. Následne sa vygeneruje k symbolov. V dalšom kroku nemôže model zostať v rovnakom stave. Jednou z nevýhod je, že pri použití Viterbiho algoritmu na tomto type modelu je jeho časová zložitosť priamo úmerná najvyššiemu možnému k, ktoré môže nastať.

Na hľadanie génov existuje veľa programov, pričom spomenieme dva súvisiace s našou prácov :

ExonHunter [2] pri svojej činnosti využíva okrem sekvencie DNA aj čo najväčšie množstvo externých informácií [2.5]. Používa zovšeobecnený model HMM podobne ako programy GENSCAN [4] alebo AUGUSTUS [12]. Hlavné rozdiely, v ktorých sa líši, sú:

- **Obsah GC párov**: je to model tranzičných a emisných pravdepodobností závisiaci od obsahu báz G a C v sekvencii. V oblastiach bohatých na gény býva obsah týchto párov zvyčajne vyšší ako v oblastiach chudobných na gény. Ostatné programy na hľadanie génov nastavujú tento parameter na rovnaké číslo pre celú sekvenciu. Exonhunter vyhodnocuje tento parameter podľa okna dĺžky 1000bp okolo aktuálnej pozície.

- **Modelovanie signálov**: autori používajú high order trees (stromy vysokých rádov) druhého rádu na modelovanie signálov donora a akceptora. Oproti ostatným modelom je to len malé vylepšenie v schopnosti rozlišovať signály, ale poskytujú presnejšie odhady pravdepodobností než iné modely.

- **Distribúcia dĺžok**: technika modelovania exónov a intrónov je vyvinutá v spolupráci Brejová a Vinař (2002). Distribúcia dĺžok je rozdelená na dve časti. Hlava môže obsahovať akúkoľvek distribúciu a chvost, ktorý geometricky klesá. Rýchlosť behu sa zmenila na $O(nd)$, kde d je dĺžka počiatočného regiónu hlavy. Pre reálne použitie v programoch sú použiteľné len malé hodnoty parametra d.

SNAP [8] je program na hľadanie génov, ktorý nevyužíva externé informácie. Na vstupe má samotné sekvencie DNA a HMM ktoré je natrénované na príslušnom organizme. Používa podobný model ako v programe Genscen [4] s rozdielmi:

- Snap používa 6 intrónových stavov na zabránenie stop kodónov v miestach zostrihov intrónov

- Genscan modeluje oba smery prekladu súčasne, ale Snap každý smer samostatne. Výhodou je, že môže nájsť gény v intrónoch iných génov.

- V genscan je stavový diagram HMM fixný, ale v Snape je diagram načítaný zo súboru.

2.5 Využitie externých informácií v hľadaní génov

Externé informácie nám pomáhajú presnejšie určiť pozíciu exónov. Jedinou nevýhodou je, že nie vždy sú všetky informácie dostupné.

Expressed sequence tag (EST) sú krátke sekvenované úseky mRNA o dĺžke 300-600 nt, pričom neobsahujú intróny. Aj keď sú len takéto krátke, stačia na určenie podobnosti so známymi alebo zatiaľ nedefinovanými génmi. Pomáhajú pri bližšom určení okrajov exónov.

Proteíny z iného organizmu môžeme tiež použiť na hľadanie génov v skúmanom organizme. Využijeme pritom predpoklad, že proteíny v príbuzných organizmoch s rovnakou funkciou budú mať podobné sekvencie aminokyselín . V tomto prípade je najlepšie využiť proteíny z organizmu, ktorý je evolučne čo najbližšie k skúmanému

organizmu. Predpokladáme, že ak sa nám podarí zarovnať proteín z iného organizmu do skúmaného genómu, tak je veľká pravdepodobnosť, že zarovnanie tiež obsahuje gén kódujúci podobný proteín.

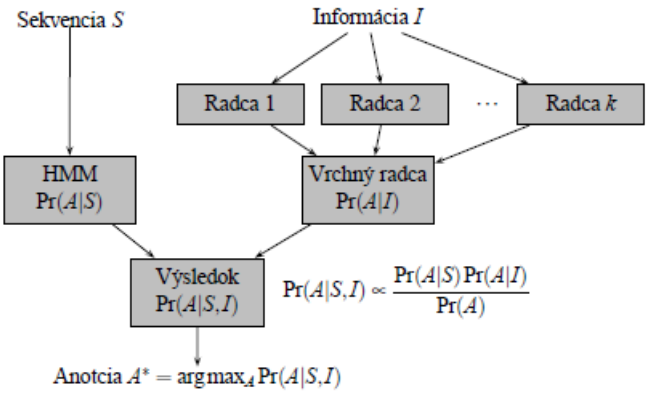
Sekvenčné opakovania sú časti DNA, ktoré sa opakujú vo veľa kópiách. Predpokladá sa, že takto opakované časti DNA väčšinou nekódujú proteíny. Poznáme viaceré typy sekvenčných opakovaní:

- tandemové opakovania sú také, ktoré sa veľakrát po sebe opakujú, napríklad sekvencia DNA: ATTGCATATTGCATATTGCAT znázorňuje 3x po sebe sa opakujúcu sekvenciu ATTGCAT. Tieto môžeme následne dalej rozdeliť podľa veľkosti opakovania na:
  - Satelitná DNA môže byť dlhá až niekoľko tisíc bázových párov
  - Minisatelity majú dĺžku 10 až 60 bázových párov
  - Mikrosatelity sú kratšie ako 10 bázových párov
- rozptýlené opakovania sú umiestnené v rôznych častiach DNA. Pomáhajú vzniku nových génov. Rozdeľujeme ich na:
  - SINE (Short Interspersed Nuclear Elements) sú opakujúce sa sekvencie DNA o dĺžke menej ako 500 bázových párov. Neobsahujú žiadne kódujúce sekvencie. Tvoria približne 11% v genóme človeka a majú približne 1,5 miliona kópií.
  - LINE (Long Interspersed Nuclear Elements) sú opakujúce sa sekvencie DNA ktoré sú dlhšie ako SINE. Tvoria približne 25% v genóm človeka a majú približne 850 tisíc kópií.

RepeatMasker
RepeatMasker je program, ktorý sa používa na vyhľadávanie rozptýlených opakovaní a DNA sekvencií s nízkou zložitosťou (low complexity DNA sequences). Na vyhľadávanie používa knižnice opakovaní. V súčasnosti sú podporované knižnice Dfam a RepBase. Samotné vyhľadávanie je založené na zarovnávaní vstupnej sekvencie ku opakovaniam z knižnice.

Externé informácie sa využívajú vo viacerých programoch na hľadanie génov. Z programov, ktoré sme používali v diplomovej práci, sú to programy ExonHunter[2] a Maker[5].

Exonhunter je program, ktorý na spracovanie externých informácií využíva radcov a vrchného radcu.



Obr. 2.7: Architektúra programu exonhunter [3]

Radca zodpovedá jednej črte z externých informácií. Jeden radca poskytuje čiastkovú informáciu o celkovom rozložení pravdepodobností cez všetky skryté stavy na danej pozícii. Radca nemusí mať informácie na vyhodnotenie všetkých symbolov z množiny H, ktorá je množinou biologických pojmov(emitovaných symbolov).

$$H = \{exón, intrón, štart\ kodón, stop\ kodón, donor, akceptor, medzigénová\ časť\}$$

Napríklad radca zodpovedajúci za zarovnanie EST ku sekvencii vyhodnotí, že na danom mieste by mohol byť exón s pravdepodobnosťou P, ktorá môže vyplývať z kvality zarovnania. Ostatné symboly z množiny H budú mať pravdepodobnosť $1 - P$. V prípade, že radca nemá žiadnu informáciu o danej pozícii, vyhodnotí všetky symboly z množiny H s pravdepodobnosťou 1, čo znamená, že na danom mieste sa môže nachádzať ktorýkoľvek zo symbolov s pravdepodobnosťou 1. Formálnu definíciu pre radcu je definovaná [2, Brejova et al.] ako:

Rada radcu r na pozícii i je partícia π_r množiny H a pravdepodobnostna distribúcia $p_r(S)$ na všetkých prvkoch partície $S \epsilon \pi_r$. Hodnota $p_r(S)$ je odhad pravdepodobnosti, že správny symbol na pozícii i je v množine S, podľa informácii dostupných pre radcu r.

Vrchný radca slúži na spojenie informácií zo všetkých radcov. Pre každú pozíciu vstupnej sekvencie S špecifikuje pravdepodobnú distribúcu pre všetky symboly. Rada vrchného radcu na jednej pozícii je pravepodobnostná distribúcia $x^* = (x_1, ..., x_n)$ na množine H, kde x_i je pravdepodobnosť daného symbolu z množiny H, keď berieme do úvahy rady všetkých radcov. Distribúcia x^* je zvolená tak, aby bola čo najbližšie každému radcovi. Vzdialenosť medzi radou a x^* je definovaná ako:

$$dist_a(x^*) = \sum_{S \epsilon \pi_a} \frac{1}{prior(g)} * (p_a(S) - \sum_{j \epsilon S} x_j)^2$$

kde $prior(s)$ je apriorná pravdepodobnosť symbolu j. Vypočítame ju ako:

$$prior(s) = \sum_{x \epsilon g} prior(x)$$

Hľadanie vrchného radcu môže byť definované ako konvexný kvadratický problém, kde minimalizujeme

$\sum_a w_a * dist_a(x^*)$

za predpokladov:

$\sum_{j \epsilon H} x_j = 1$
a

$x_j \geq 0$ pre všetky symboly $j \epsilon H$.

Člen w_a slúži na váhovanie vzdialenosti medzi radcom a výslednou mierou.

Vrchný radca môže byť vypočítaný jednoduchšiou metódou založenou na lineárnej kombinácii všetkých radcov popísanú nižšie. Na každej pozícii, kde radcovia predpovedajú prázdny výsledok (výsledok, kde pre všetky symboly súčasne je predpovedaná pravdepodobnosť 1), bude vymazaný. Výsledná pravdepodobnostná distribúcia vrchného radcu je vypočítaná lineárnou kombináciou týchto jednoduchých predpovedí radcov.

Výsledná postupnosť symbolov vzniká za pomoci Viterbiho algoritmu, ktorý je ovplyvnený radami x^* vytvoreným vrchnými radcom. HMM definuje pravdepodobnosť $Pr(S|X)$, čo je pravdepodobnosť, že HMM vygeneruje postupnosť stavov S na základe vstupnej sekvencie X. Obdobne vrchný radca definuje $Pr(S|I)$, kde I je externá informácia. Budeme hľadať najpravdepodobnejšiu postupnosť symbolov, ktorú môžeme dostať súčasne zo vstupnej sekvencie a externých informácií $Pr(S|X, I)$. Túto môžme definovať podľa Bayesového pravidla ako:

$$Pr(S|X, I) = \frac{Pr(X, I|S) * Pr(S)}{Pr(X, I)}$$

Za predpokladu, že externé informácie a informácie obsiahnuté vo vstupnej sekvencii sú navzájom nezávislé, môžeme zjednodušiť. Tento predpoklad nie je pravdivý, ale snažíme sa limitovať závislosť použitím rôznych čŕt sekvencií v HMM a v radcoch.

$$Pr(S|X, I) = Pr(A|X) * \frac{Pr(S|I)}{Pr(S)}$$

Podobne ExonHunter predpokladá, že stavy nachádzajúce sa na rôznych pozíciach sú navzájom nezávislé, a teda môžeme pravdepodobnosť $P(S|I)$ vypočítať ako súčin rád vrchného radcu pozíciu po pozícii.

Maker[5] využíva zreťazené spracovanie (pipeline) dát. Jeho beh sa dá rozdeliť do 5 častí:

1. *Výpočet* - V tomto kroku sa spúštajú jednotlivé programy na prípravu vstupných dát. Na nájdenie všetkých sekvenčných opakovaní používa RepeatMasker. Tieto sa následne vylúčia z tvorby BLAST zarovnania ale ponechajú sa dostupné na zaradenie do anotácie. V druhom kroku Maker používa vlastnú databázu transpozómov a vírusov kódujúcich proteíny na identifikovanie pohyblivých elementov, ktorú zarovnáva za pomoci BLASTX.

2. *Filtrovanie a Klastrovanie* - Tento krok slúži na odstránenie marginálnych predikcií za pomoci skóre, percentuálneho pokrytia a.i. Kritéria pre filtrovania sa načítavaju zo súboru (maker bopts.ctl). Klastrovanie slúži na zlúčenie dát do skupín, v ktorých všetky podporujú rovnaký gén alebo transkript a vyraďuje nadbitočné informácie. Napríklad pre jeden úsek DNA existuje veľké množstvo zarovnaných EST, do konfiguračného súboru je možné nastaviť maximálny počet použitých EST prechádzajúcich do dalších krokov.

3. *Čistenie* - V tomto kroku sa porovnajú výsledky z BLAST-u a druhého zarovnávajúceho algoritmu (Exonerate) na získanie presnejších okrajov exónov. Súčasne Exonerate poskytuje informácie o donoroch a akceptoroch.

4. *Syntéza* V tomto kroku sa využívajú informácie vytvorené počas filtrovania, klastrovania, čistenia a zarovnania proteínov na vytvorenie anotácie.

   Maker najprv určí typ daného nukleotidu na základe všetkých informácií. Možné priradenia sú: Medzigénový, kódujúci, intron a UTR. Napríklad prekryvu EST a zarovnania, v prípade že sa zhodujú, je to pravdepodobne kódujúci nukleotid ak nie, medzigénový nukleotid. Maker vypočíta skóre pre každý nukleotid na základe podobnosti zarovnania, typu zarovnania a jeho pozícii v zarovnaní. Na základe týchto informácii SNAP upravý interný HMM model.

5. Anotácia - Záverečná úprava SNAP predikcií voči všetkým informáciam. Koordináty sú upravené tak, aby zahŕňali informácie o EST, mRNA, 5', 3' a UTR.

2.6 Trénovanie modelov pre hľadanie génov

Cieľom trénovania modelov pre hľadanie génov je zvýšiť presnosť predikcie génov. Tomuto môže pomôcť použitie externých informácií. Trénovanie modelov je veľmi závislé na na tom, koľko externých informácií máme o danom organizme a v prípade, že používame informácie z iného organizmu, tak aj fakt, ako veľmi je evolučne vzdialený od hľadaného organizmu. Externé informácie, ktoré môžeme mať k dispozícii a sú z rovnakého organizmu, sú EST, PET. Na zlepšenie predikcie génov môžeme použiť aj informácie z iného organizmu ako proteíny, zarovnania ku genómom alebo databázu repeatov.

Na natrénovanie modelu nemáme k dispozícii správne riešenie, lebo sa sažíme nájsť správnu anotáciu nového organizmu, ktorý nebol ešte anotovaný. Pre prvý beh programu používame natrénovaný model (HMM) na inom organizme, pričom sa snažíme, aby tento iný organizmus bol čo evolučne najbližšie. Výsledkom tohoto behu je anotácia, ktorá nie je presná. Z toho dôvodu chceme za pomoci externých informácií vybrať z anotácie tie o ktorých máme predpoklad, že patria do správnej anotácie. Výsledkom tohoto je podmnožina pôvodnej anotácie, kde sa nachádza väčší pomer génov (true positive) voči oblastiam, ktoré génmi nie sú (false positive). Na základe tejto podmnožiny upraviť HMM model pre program hladajúci gény. Následne spustiť program s novým HMM modelom na pôvodných dátach. Výsledkom po prvej iterácii je presnejšia anotácia daného organizmu. Počet iterácií sa dá opakovať až do momentu kím sa výsledky budú zlepšovať

Podstatnou časťou tohto problému, ktorú riešime aj v našej diplomovej práci, je ako vybrať podmnožinu génov na natrénovanie modelu. Tento problém ovplyvňuje množstvo externých informácií, ktoré sú k dispozícii a akú váhu priradiť tej ktorej externej informácii.

Iné programy využívajúce trénovanie modelov sú napríklad:
Exonhunter[2]

Program Exonhunter má novú iteratívnu techniku, ktorá berie do úvahy viaceré externé informácie. Pri použití externej informácie je tento program presnejší ako v prípade, že by túto informáciu nepoužil. Kvôli tomuto je tento prístup vhodnejší na vytvorenie trénovacieho setu pre programy na hľadanie génov ako *ab initio*(predikcie bez použitie

externých informácií) predikcie.

Architektúra programu ExonHunter pozostáva z dvoch základných častí HMM a radcov [bližšie viď 3.3 ExonHunter]. Výstupné informácie z oboch častí sú kombinované, čím je ovplyvnený Viterbiho algoritmus, výsledkom, ktorého je anotácia.

Na identifikáciu podporovaných fragmentov klasifikovali každú pozíciu ako:

- *podporovaná* - bola označená v prípade, že predikovaný symbol mal najvyššie skóre spomedzi všetkých symbolov. Naviac musel spĺnať podmienku, že veľkosť skóre musela byť väčšia ako predom určená hranica(používaná 1.01)
- *konfliktná* - bola označená v prípade, že nejaký iný symbol na tejto pozícii by bol označený ako podporovaný.
- *neznáma*

Intrón a exón sa považuje za podporovaný v prípade, ak má aspoň polovicu báz označenú ako podporované a neobsahuje žiadnu bázu označenú ako konfliktnú. Z toho vyplýva, že na označenie intrónu alebo exónu za konfliktný stači označit jednu jeho bázu ako konfliktnú. V prípade, že intrón alebo exón nespĺňa ani jednu z predošlých dvoch podmienok, je označený ako neznámy.

V ďalšom kroku spája podporované exóny do dlhších reťazí, ak sú medzi nimi intróny, ktoré nie sú konfliktné. Každá z reťazí je použitá ako potenciálne nekompletný transkript vo filtrovanej trénovacej množine.

Na určenie parametrov HMM použili takéto zvolené fragmenty, okrem medzigénových, ktorých distribúcia je odvodená z úplných nefiltrovaných predpovedí. Nefiltrované predikcie sú využité aj na trénovanie apriornej distribúcie vrchného radcu a všetkých ostatných parametrov vrchného radcu. Použitie týchto fragmentov by mohlo viesť k podhodnoteniu apriorných pravdepodobností exónov a intrónov.
HMM je trénované pomocou Viterbiho trénovania. Viac o tomto probléme 2.6

HMMGene [10] a GENE Kulp et al., 1997) - programy na hľadanie génov, ktoré využívajú externé informácie, ale ich nevýhodou je, že v jednom miete sekvencie môže využívať len jeden druh externéj informácie. Exonhunter kombinuje zdroje viacerých externých informácii za pomoci kvadratického programovania (alebo v našom prípade lineárnou kombináciou).

EuGène [17] - založený na pravdepodobnostne motivovaných acyklických grafoch. Využíva viaceré externé informácie súčastne (EST, proteíny). Externé informácie priamo modifikujú hrany grafu.

Combiner [1] - kombinuje predpovede niekoľkých programov na hľadanie génov a sekvenčných zarovnaní. Na skĺbenie používa rozhodovacie stromy a dynamické programovanie. Prístup programu Exonhunter je lepši.

Kapitola 3

Výber trénovaných dát na novom genóme

V tejto kapitole popíšeme náš prístup na výber predpokladaných génov, ktoré budú slúžiť na natrénovanie HMM v programe Snap. Vytvorili sme viaceré kritériá, za pomoci ktorých sme z množiny predpovedaných génov vybrali tie, ktoré na základe externých informácií majú väčšiu šancu byť správne.

3.1 Dáta

V tejto podkapitole popíšeme, aké vstupné dáta používame.

Zarovnávali sme genóm organizmu *D.melanogaster*. Vstupnú DNA sekvenciu sme mali rozdelená na neprekrívajúce sa množiny:

- **Trénovacia množina** Obsah tejto mnoožiny je približne 32% zo všetkých vstupných dát. Bola použitá na natrénovanie vlastného spôsobu výberu génov. Toto sme realizovali trénovanim SVM [viac v kapitola 5].
- **Validačná množina** Obsah tejto mnoožiny je približne 18% zo všetkých vstupných dát. Slúžila na krosvalidáciu trénovacej množiny pri trénovaní podporných vektorov pre SVM model, čo nám umožnilo lepší výber parametrov a typu SVM.
- **Testovacia množina** Obsah tejto mnoožiny je približne 28% zo všetkých vstupných dát. Bola použitá na vyhodnotenie jednotlivých metód výberu dát pri rôznych nastaveniach parametrov. Najlepšie spôsoby boli vybrané na natrénovanie HMM modelu pre SNAP

- **Testovaciu množina 2** Obsah tejto mnoožiny je približne 22% zo všetkých vstupných dát. slúžila na záverečné testovanie natrénovania HMM modelov pre program SNAP

Každá z týchto množín obsahuje DNA sekvenciu a správnu anotáciu na kontrolu alebo trénovanie.

Externé informácie:

- **EST** boli z rovnakého organozmu D.melanogaster. Obsahuje 55350 EST o celkovej dĺžke 74248261 bázových párov.
- proteíny su z organizmu Apis mellifera. Obsahuje 10910 proteínov o celkovej dĺžke 5893710 kodónov.
- opakovania sme dostali za pomoci zarovnania jednotlivých množín ku knižnici opakovaní progranu RepeatMasker. Databáza opakovaní bola použitá z organizmu D.melanogaster, ktorá obsahuje 65 opakovaní o celkovej dĺžke 167423 bázových párov

3.2 EST

Porovnávali sme predpovedané gény s EST sekvenciami, pričom predpokladáme, že exóny, ktoré sa silne podobajú na niektoré EST, majú väčšiu šancu byť správne.
Ako vstup sme mali EST sekvencie vo formáte fasta. Tento formát pozostáva z identifikátora danej EST sekvencie a následnej sekvencie, ktorá je rozdenená po 60 znakov na riadok.
Túto sekvenciu sme lokálne zarovnali za pomoci programu BLAT voči jednotlivým predpovedaným génom. Výstupom z tohoto programu je tabuľka vo formáte psl, ktorá obsahuje:

- identifikátor zarovnávanej a zarovnávajúcej sa sekvencie
- pozíciu a dĺžku zarovnávanej a zarovnávajúcej sa sekvencie
- informáciu o tom, aké dobré je zarovnanie (match, mismatch, gap)
- ostatné

Túto tabuľku sme následne spracovali a hľadali gény, ktoré boli pokryté aspoň jedným EST tak, aby spĺňali podmienky:

- EST zarovnané vo vnútri génu musí mať zarovnané aspoň 95% svojej dĺžky. Je to z dôvodu, že EST pochádza z mRNA pre určitý gén, mala by sa s jeho časťou zhodovať, pričom dovoľujeme určitú toleranciu kvôli sekvenovacím chybám.
- V prípade, že sa EST zarovnanie nachádza na konci alebo začiatku génu, nemusí pokrývať celé EST, nakoľko EST môže zahŕňať neprekladané konce mRNA, ktoré predpovedané gény neobsahujú. Pri určovaní, či je zarovnanie na začiatku génu, povoľujeme toletanciu 5% vyplývajúcu z toho, že dané zarovnanie nemusí mať úplne presné hranice. Tieto chyby zodpovedajú sekvenovacím chybám pri získavaní EST a tie by nemali byť príliš početné.
- maximálna chyba zarovnania bude 5% z jeho dĺžky. Snažíme sa nájsť len najlepšie zarovnania, ktoré sú k dispozícii.

Ako voliteľný vznikol další parameter - percento pokrytia génu aspoň jedným zarovnaným EST. Tento parametr vyjadruje koľko báz z exónu máme pokrytých aspoň jedným EST. Ak bolo percento vyššie ako tento parameter, prehlásili sme gén ako predpokladaný.

Proces hľadania zarovnaných génov je prechádzanie cez všetky gény, ktoré majú aspoň jedno EST zarovnanie. Následne pre každý takýto gén vypočítame percento zarovnania, a ak je vyššie ako vstupný parameter, prehlásime gén za predpokladaný.

	SNAP	100 % EST	80% EST	60% EST	40% EST
Exon Sensitivity	18.50%	12.67%	17.54%	18.16 %	18.36 %
Exon Specificity	30.68%	47.03%	46.56%	44.82%	43.36%

Tabuľka 3.1: Tabuľka výsledkov pri použití EST

V tabuľke 3.1 jednotlivé stĺpce predstavujú jednotlivé nami vyfiltrované množiny génov.

SNAP - Vstupné dáta sú výsledkom zarovnania za pomoci programu SNAP. Pri zarovnaní bolo použité HMM z organizmu *C.elegans*. Jednotlivé výsledky niesu dobré z dôvodu že simulujeme prípad zarovnania na novom organizme pre ktorý neexistujú informácie otom ako správne zarovnanie vypadá. Snažíme sa vyfiltrovať čo najviac nesprávnych génov z tejto množiny a pritom ponechať čo najviac správnych génov.

X% EST - jednotlivé vyfiltrované množiny génov pri nastavení parametru 'percento pokrytia génu' na hodnotu X.

Jednotlivé riadky predstavuje všeobecná metrika bližšie vysvetlená v kapitole 2.2.

Z tabuľky je vidno, že ak vybereme len zarovnané EST ku vstupnej DNA sekvencii, tak sa nám podarí vyfiltrovať časť falošne pozitívnych exónov a popritom zachovávame skoro všetky skutočne pozitívne exóny.

3.3 Proteíny

V tomto experimente porovnávame predpovedané gény s proteínmi z iného organizmu, pričom predpokladáme, že ak nájdeme zarovnanie s proteínom tak zarovnaná časť génu má väčšiu pravdepodobnosť byť správna.

Na lokálne zarovnanie známych proteínov a predpovedaných proteínov sme použili program BLAST. Výstup z tohoto programu je však chudobnejší na dáta ako výstup z programu BLAT, ale má výhodu zarovnania už preložených proteínov:

- identifikátor zarovnávanej a zarovnávajúcej sa sekvencie
- pozícia zarovnávanej a zarovnávajúcej sa sekvencie
- informáciu o tom, aké dobré je zarovnanie (e-value, bit-score)

Pri spracovaní dát sme potrebovali aj surové dáta, aby sme vedeli vypočítat presné dĺžky génov, ktoré nieboli zahrnuté vo výstupe z programu BLAST. Takto vznikli kritériá:

- **Bit score** - určuje, aké dobré je dané zarovnanie.
- **E-value** - vyjadruje pravdepodobnosť, že existuje iné zarovnanie, ktoré bude zarovnané s väčším bit score. Pre tento parameter sme určili hodnotu $1e^{-}5$ pri hľadaní zarovnaní za pomoci programu BLAST. To znamená, že na jeho výstupe budú len zarovnania, ktorých e-value bude niŠšie ako $1e^{-}5$.
- Percento pokrytia predpokladaného proteínu proteínmi z Drosophily.

  Algoritmus na nájdenie všetkých génov, ktoré spĺnaju dané kritériá je, že prechádzame cez všetky zarovnania proteínov a v prípade, že toto zarovnanie pokrýva aspoň parameter 'percento pokrytia predpokladaného proteínu', tak ho prehlásime za predpovedaný gén.

	SNAP	100 % prot	80 % prot	60% prot	40% prot
Exon Sensitivity	18.50%	0.04%	1.02%	1.36%	1.54%
Exon Specificity	30.69%	42.86%	50.59%	51.58%	50.59%

Tabuľka 3.2: Tabuľka výsledkov proteínov

Tabuľka 3.2 má rovnaké rozloženie ako tabuľka použitá pri EST.

Stĺpce **X%prot** sú jednotlivé vyfiltrované množiny pri nastavení parametru 'percento pokrytia predpokladaného proteínu' na hodnotu X

Z tabuľky vyplýva, že nájsť 100% zarovnaný proteín je veľmi ťažké. Ale proteíny, ktoré sú zarovnané s DNA sekvenciou, sú v špecificite lepšie ako zakladná predikcia SNAP-u.

Kapitola 4

Využitie strojového učenia

V postupe, ktorý uplatňujeme, sa nachádza veľké množstvo parametrov, ktoré musíme nastavovať. Ručne nevieme nastaviť všetky parametre na najlepšiu hodnotu, preto chceme použiť metódu, ktorá by na trénovacích dátach vedela nastaviť tieto parametre a následne ich mať už nastavené na hľadanie génov v inom organizme. Súčasne chceme zlepšiť doterajšie výsledky pridávaním väčšieho množstva čŕt.

Na klasifikáciu je možné použiť viaceré metódy ako SVM (support vector machine), logická regresia alebo neurónové siete. V tejto práci sme použili SVM.

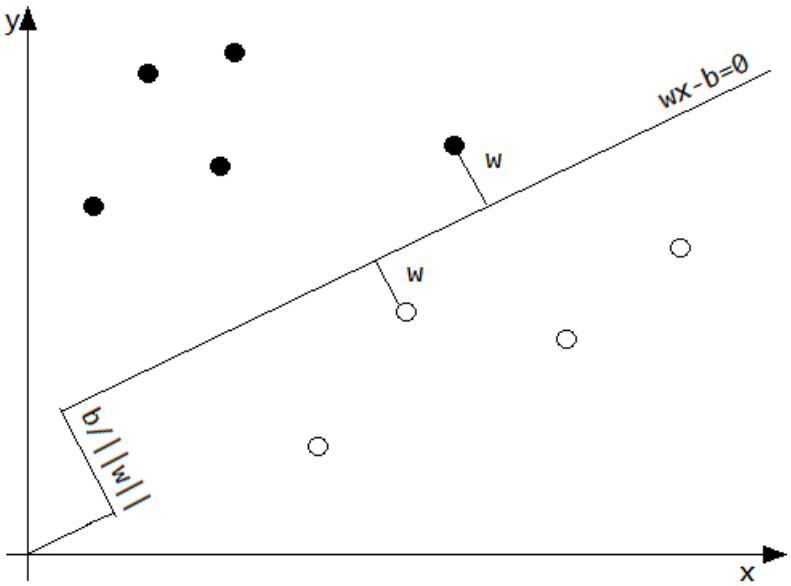
4.1 Úvod do SVM

Snažíme sa násť nadrovinu, ktorá má čo najväčšiu vzdialenosť od trénovaných príkladov ako v obrázku 4.1. Vstupom na trénovanie pre SVM sú vstupné atribúty $x \epsilon R^n$ a cieľové premenné $y_i \epsilon \{-1, 1\}$

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{in} \end{pmatrix}$$

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \end{pmatrix}$$

Najprv sa musí SVM natrénovať na trénovacích dátach. To zanmená vypočítať $\lambda = (\lambda_1, ..., \lambda_i)$ také, že maximalizujú



Obr. 4.1: Príklad maximalizovania vzdialenosti v SVM

$\sum_{i=1}^{t} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j K(x_i, x_k)$ za podmienok $(\forall i) \lambda_i \geq 0$

Na vypočítanie použijeme solver kvadratických programov.

Vypočítame $w = \sum_{i=1}^{t} \lambda_i y_i x_i$

Vyberieme dva vektory, ktoré sú v minimálnej vzdialenosti k nájdenej nadrovine, z ktorých jeden sa bude nachádzať v $x_i y_i = +1; \lambda_i > 0$ a druhý v $x_j y_j = +1; \lambda_j > 0$. Pomocou nich vzpočítame b.

$b = \frac{1}{2} \sum_{k=1}^{t} \lambda_i y_i (K(x_k, x_i) + K(x_k, x_j))$

Druhým krokom je klasifikácia testovacích dát:

$\hat{y} = sgn(< w, x > -b) = sgn([\sum_{i=1}^{t} \lambda_i y_i K(x_i x)] - b)$

V tejto metóde sa využíva **Kernelova fukcia** $K(X, Z)$. Jej intuitívny význam je miera podobnosti medzi vektormi X a Z. V prípade, že sú X a Z podobné $K(X, Z)$ je veľke, v opačnom prípade sa hodnota K(X,Z) blíži k nule.

$K(X, Z)$ vypočítame tak, že rozvinieme jednotlivé atribúty pomocou bázových funkcií φ, to počítame cez skalárny súčin $K(X, Z) =< \varphi(X), \varphi(Y) >$

Výhoda použitia kernelu je, že nemusíme robiť rozvoj bázových funkcií a počítame len cez skalárny súčin X a Z.

V prípade, že spravíme úplný rozvoj kernelovej funkcie pre dimenziu d, budú výsledkom všetky polynómy stupňa nanajvýš d. Táto funkcia sa nazýva **polynomiálny kernel**.

Vektory X a Z nemusia predstavovať číselný vektor v R^n, môžu to byť obrázky alebo texty. Musí byť zachovaná len podmienka, že kernel vyjadruje podobnosť medzi X a Z

Najpoužívanejšie kernely sú:

- **Lineárny** $K(X, Z) = X^T Z$
- **Polynomiálny** $K(X, Z) = (X^T Z + c)^d$ je polinomiálny kernel stupňa d
- **Gausovský** $K(X, Z) = exp(-\frac{||X-Z||^2}{2\delta^2}$
- **RBF** radial basis function $K(X, Z) = exp(-\gamma * ||X - Z||^2)$
- **Sigmoidálny** $K(X, Z) = \tanh(\alpha X^T Z + c)$

Trénovaním SVM sa zaoberajú viaceré hotové riešenia ako napríklad:

- **Libsvm** - je vyvinutý v National Taiwan University.
- **SVM Light** - je vyvinutý v Cornell University.
- **Shogun** - je vyvinutý v Fridrich Miescher Laboratory Tübingen a je nadstavbou na predchádzajúce dve riešenia.

4.2 Vytvorenie trénovacích dát pre SVM

Hlavnou časťou je vytvorenie jednotlivých čŕt (z ang. feature), ktoré budú použité na trénovanie. Tieto črty sú vytvorené z externých informácií. Dáta na experiment sú rovnaké ako dáta v časti 3.

Zo zarovnaní predpovedaných génov a EST sme vytvorili dve črty, ktoré charakterizujú zarovnanie:

1. Percento predpovedaného exónu, ktoré je zarovnané s aspoň jedným EST.
2. Chyba pokrytia zarovnaných EST k predpovedanému génu.

Zo zarovnaní predpovedaných génov ku proteínom sme vytvorili štyri črty:

1. Percento predpovedaného exónu, ktoré je zarovnané s aspoň jedným proteínom z Drozofili.

2. Bit skóre zarovnania najlepšieho proteínu zarovnaného ku predpovedanému génu, ktorú sme vydelili dĺžkou daného zarovnania. Vydelenie sme urobili z dôvodu, že dlhé zarovnanie bude mať vyššie skóre ako krátke, aj keď ich kvalita môže byť rovnaká.

3. E-value zarovannia pre najlepšie bit skóre.

4. Pokrytie krajov je parameter, ktorý určuje, či daný exón je zarovnaný s proteínom v strede zarovnania, na kraji alebo po celej dĺžke. Z tohoto vyplýva, že výsledná hodnota bude počet zarovnaných krajov (0,1,2). Kraj je braný s 5% toleranciou z dôvodu, že okraje exónu nemusia byť určené presne.

Ďalšími pridanými dátami je databáza hľadaných repeatov (časti DNA, ktoré, ktoré sa opakujú). Hľadali sme prekryvy aspoň dĺžky 10 medzi exónmi v predikovaných génoch a databázou repeatov. Z tejto externej informácie sme vytvorili jeden parameter. Ak predikovaný exón má prekryv s repeatom, nastavili sme mu parameter na 1, v ostatných prípadoch sme tento parameter nastavil na 0. Táto informácia nám pomáha určit nesprávne predikované exóny, nakoľko repeaty sa nenechádzajú v kódujúcich častiach genómu.

Beh programu pozostáva z vygenerovania čiastkových trénovacích(testovacích) množín pre každú externú informáciu zvlášť. Táto množina je v súbore zapísaná vo formáte ako na obr. 4.2. Tento formát bol zvolený kvôli kompatibilite a ľahkému prevodu na formát formátu pre použitie SVM. Prvé tri riadky sú informácie o počte čŕt, ich názvov a defaultných hodnôt. Defaultné hodnoty sú použité v situácii, keď spájame jednotlivé trénovacie (testovacie) množiny a niektorý z exónov nemá zastúpenie nijakou črtou. Nasledujú jednotlivé trénovacie (testovacie) príklady pre SVM, na ktorých začiatku je identifikátor exónu používajúci sa na spárovanie.

Druhým krokom je spojenie všetkých trénovacích množín s rôznymi typmi informácie do jednej. Pri tomto kroku algoritmus dopĺňa v príkladoch nenachádzajúce sa hodnoty defaultnými hodnotami. Toto sa deje z dôvodu, že algoritmus SVM sa veľmi ťažko vyrovnáva s chýbajúcimi hodnotami v jednotlivých príkladoch.

Posledným krokom je v trénovacej sade nahradenie identifikátora exónu za skutočnú hodnotu či je daná sekvencia DNA je exón. Pri tomto kroku je nutné dbať na presnú

```
% 1:prot_pokritie 2:bitScore/length 3:e-value 4:pokritie_krajov
% 1:0 2:0 3:0 4:0
chrI-5977985.3.1.17 1:1.0 2:1.33 3:-390.74631862842784 4:1
chrI-5977985.3.1.18 1:100.0 2:1.33 3:-390.74631862842784 4:2
chrI-5977985.3.1.19 1:100.0 2:1.33 3:-390.74631862842784 4:2
chrI-5977985.3.1.20 1:100.0 2:1.33 3:-390.74631862842784 4:2
chrI-5977985.3.1.21 1:71.0 2:1.33 3:-390.74631862842784 4:1
chrI-5977985.5.1.1 1:65.0 2:1.07 3:-15.424948470398375 4:1
chrI-5977985.5.1.2 1:97.0 2:1.07 3:-15.424948470398375 4:1
chrI-5977985.5.1.8 1:55.0 2:1.07 3:-15.424948470398375 4:1
```

Obr. 4.2: Príklad použíteho formátu pre trénovacie a testovacie množiny

zhodu oboch okrajov exónu medzi predikciou a správnou anotáciou. Výsledný formát je možné vidieť na obrázku 4.3. SVMLight bude ignorovať naše dodatočné informácie, nakoľko znak % odignoruje ako poznámku.

```
prot_pokritie 2:bitScore/length 3:e-value 4:pokritie_krajov 5:est_pokritie 6:est
0 2:0 3:0 4:0 5:0 6:100 7:0
I-1993048.36.1.4 1:0.0 2:0.0 3:0.0 4:0.0 5:100.0 6:23.0 7:0
I-1993048.36.1.5 1:0.0 2:0.0 3:0.0 4:0.0 5:100.0 6:23.0 7:0
I-1993048.36.1.6 1:0.0 2:0.0 3:0.0 4:0.0 5:50.0 6:23.0 7:0
I-1993048.36.1.0 1:0.0 2:0.0 3:0.0 4:0.0 5:100.0 6:23.0 7:0
I-1993048.36.1.1 1:0.0 2:0.0 3:0.0 4:0.0 5:100.0 6:23.0 7:0
I-1993048.36.1.2 1:0.0 2:0.0 3:0.0 4:0.0 5:100.0 6:23.0 7:0
I-1993048.36.1.3 1:0.0 2:0.0 3:0.0 4:0.0 5:100.0 6:23.0 7:0
II-3997309.189.1.8 1:0.0 2:0.0 3:0.0 4:0.0 5:94.0 6:1.0 7:0
```

Obr. 4.3: Príklad použíteho formátu pre SVM

4.3 Experimenty

Na klasifikáciu dát sme použili knižnicu SVM-light [7].

Na trénovanie sme použili iteratívny proces. Na trénovacej množine sme natrénovali SVM model (pozostáva z podporných vektorov a vypočítaného parametru b) s počiatočnými parametrom. Následne sme klasifikovali dáta v validačnej množine a vypočítali percento správne určených prvkov. Nasledovala úprava parametra pre SVM model. Tento proces sme opakovali až do chvíle, keď percento správne určených prvkov nezačalo klesať. V našich experimentoch sme použili RBF kernel s parametrom c nastaveným na 0.90. Parameter γ sme nechali nastavený na defaultnú hodnotu. Tento parameter určuje kompromis medzi trénovacou chybou a marginom.

Po natrénovaní modelu je výstupom súbor obsahujúci zoznam podporných vektorov, b a parametre, pri ktorých bol model natrénovaný.

Výstupom klasifikácie je zoznam príkladov v rovnakom poradí ako bol na vstupe. Ku každému príkladu z testovacej množiny je priradená predikcia. Aby sme určili hranicu, ktorá bude rozdeľovať príklady na predpovedané gény a nepredpovedané, v jednoduchom prípade stačí použiť funkciu $sgn(x)$, ktorej výsledok je znamienko hodnoty x.

	SNAP	Dáta z SVM	EST	Proteíny
Exon Sensitivity	18.50%	4.93%	18.36%	1.54%
Exon Specificity	30.69%	49.11%	43.36%	50.59%

Tabuľka 4.1: Tabuľka výsledného algoritmu používajúceho SVM a

Z tabuľky 4.1 je vidno, že daná metóda našla väčšie množstvo exónov ako samostatné proteíny a zároveň našlo lepší pomer skutočne pozitívnych ku falošne pozitívnym ako v prípade použitia samostatných EST.

4.4 Trénovanie HMM modelu

V tejto podkapitole sa budeme venovať spôsobu a výsledkom trénovania HMM modelu pre program SNAP.

Na trénovanie HMM modelu pre program SNAP je za potrebné mať oanotovanú sekvenciu DNA, v ktorej sú určené exóny start kodóny a stop kodóny. Pre samotné trénovanie je potrebné previesť anotáciu vo formáte gtf na formát zff.

```
1  >2L-2680
2  Einit 37441 37462 2L-2680-snap.10
3  Exon 36622 37178 2L-2680-snap.10
4  Eterm 35852 36052 2L-2680-snap.10
5  Einit 71849 71893 2L-2680-snap.15
6  Exon 72224 72339 2L-2680-snap.15
7  Exon 72399 72987 2L-2680-snap.15
8  Exon 73725 74735 2L-2680-snap.15
9  Exon 74802 74904 2L-2680-snap.15
10 Eterm 74963 75030 2L-2680-snap.15
11 Esngl 145637 146125 2L-2680-snap.38
```

Obr. 4.4: Príklad súboru vo formáte zff

Súbor pozostáva z identifikátora časti génu *2L-2680*. Následne na každom riadku po jednom trénovacom príklade. Prvá povinná hodnota predstavuje typ úseku. Základné hodnoty, ktoré môže nadobúdať:

- **Einit**: Prvý exón génu. V prípade že Einit začína v poradí neskôr ako Eterm, program automaticky usúdi že preklad exónu je na reverznom vlákne (zápornom). Toto je napríklad *2L-2680-snap.15* z obrázku 4.4
- **Eterm**: Posledný exón génu. V prípade že Eterm začína v poradí neskôr ako Einit, program automaticky usúdi že preklad exónu je na reverznom vlákne(zápornom). Toto je napríklad *2L-2680-snap.10* z obrázku 4.4
- **Exon**: Predstavuje exón v prípade že v jednom géne sa nachádza viac exónov oddelených intrónmi. Toto je napríklad *2L-2680-snap.15* z obrázku 4.4
- **Esgnl**: Predstavuje gén ktorý pozostáva len z jedného exónu Toto je napríklad *2L-2680-snap.38* z obrázku 4.4

Ďalšie dva stĺpce predstavujú počiatočnú a koncovú súradnicu príslušného exónu. A posledným stlpcom je nepovinný parameter - názov príkladu.
Súčasťou softwéru SNAP sú aj programy, ktoré na základe zff súboru určia parametre HMM modelu .

	C.elegans.hmm	D.melanogaster.hmm	svm.hmm	est.hmm
Exon Sensitivity	18.79%	40.54%	35.08%	30.48%
Exon Specificity	33.69%	65.24%	42.87%	50.31%

Tabuľka 4.2: Tabuľka výsledných modelou hmm

Výsledky v tabuľke boli vyhodnocované na testovacej množine 2. Prvý stĺpec predstavuje program SNAP s HMM modelom natrénovaným na organizme C.elegans. Tento výsledok je najhorší, lebo tento model bol natrénovaný len na príbuznom organizme bez použitia externých informácií. Druhý stĺpec je najlepší, pretože tento model je natrénovaný na danom organizme ak poznáme správne anotácie génov. Tretí stĺpec predstavuje HMM model natrénovaný z dát, ktoré boli vyfiltrované za pomoci SVM predchádzajúcej časti. Daný HMM model je lepši aj v specificite aj senzitivite oproti modelu, z ktorého vychádzal (model C.elegans). Posledný stĺpec predstavuje HMM model natrénovaný na jednoduchých dátach vyfitrovaných len za pomoci EST. Z tabulky je vidno, že externé informácie pomáhajú v presnejšom určení správnych génov a tieto gény následne môžu slúžiť na natrénovanie HMM modelu pre program SNAP.

Kapitola 5

Záver

Cieľom tejto práce bol návrh nového spôsobu výberu trénovacej množiny pre skrytý Markovov model, ktorý by sa dal použiť na zlepšenie kvality nájdených génov novosekvenovaných genómov. Vytvorili sme klasifkátor založený na SVM (support vextor machine), ktorý rozpoznáva v sade automaticky predikovaných génov tie, ktoré majú väčšiu pravdepodobnosť byť správne na základe externej informácie o géne. Túto metódu sme overili na genóme modelového organizmu D.melanogaster a výsledky sme porovnali s HMM modelom natrénovanom na danom organizme. Naša metóda výberu génov je lepšia ako samostatné použitie HMM modelu z iného príbuzného organizmu. Do budúcnosti plánujeme otestovať celý iteratívny proces hľadania génov s našou metódou filtrovania génov na inom organizme. Ďalšími možnosťami zlepšenia je vytvoriť dalšie črty pre trénovanie SVM, posúvanie okrajov exónov tak, aby lepšie sedeli s dostupnými externými informáciami a tým zväčšiť výsledný počet správnych exónov v trénovacej množine.

Literatúra

[1] Jonathan E Allen, Mihaela Pertea, and Steven L Salzberg. Computational gene prediction using multiple sources of evidence. *Genome Research*, 14(1):142–148, 2004.

[2] B. Brejová, D. G. Brown, M. Li, and T. Vinař. ExonHunter: A comprehensive approach to gene finding. *Bioinformatics*, 21(Suppl 1):i57–i65, 2005. Proceedings of the 15th International Conference on Inteligent Systems for Molecular Biology (ISMB 2005).

[3] Broňa Brejová, Tomáš Vinař, Yangyi Chen, Shengyue Wang, Guoping Zhao, Daniel G. Brown, Ming Li, and Yan Zhou. Finding genes in Schistosoma japonicum: annotating novel genomes with help of extrinsic evidence. *Nucleic Acids Research*, 37(7):e52, 2009.

[4] Karlin S Burge C. Prediction of complete gene structures in human genomic dna. 268:78–94, 1997.

[5] Brandi L Cantarel, Ian Korf, Sofia MC Robb, Genis Parra, Eric Ross, Barry Moore, Carson Holt, Alejandro Sánchez Alvarado, and Mark Yandell. Maker: an easy-to-use annotation pipeline designed for emerging model organism genomes. *Genome research*, 18(1):188–196, 2008.

[6] D. DeCaprio, J. P. Vinson, M. D. Pearson, P. Montgomery, M. Doherty, and J. E. Galagan. Conrad: gene prediction using conditional random fields. *Genome Research*, 17(9):1389–1398, 2007.

[7] T. Joachims. Estimating the generalization performance of a SVM efficiently. In *International Conference on Machine Learning*, pages 431–438, San Francisco, 2000. Morgan Kaufman.

[8] Ian Korf. Gene finding in novel genomes. *BMC Bioinformatics*, 5:59, 2004.

[9] Ian Korf, Paul Flicek, Daniel Duan, and Michael R Brent. Integrating genomic homology into gene structure prediction. *Bioinformatics*, 17(suppl 1):S140–S148, 2001.

[10] Anders Krogh. Using database matches with hmmgene for automated gene detection in drosophila. *Genome Research*, 10(4):523–528, 2000.

[11] Alexandre Lomsadze, Vardges Ter-Hovhannisyan, Yury O. Chernoff, and Mark Borodovsky. Gene identification in novel eukaryotic genomes by self-training algorithm. *Nucleic Acids Res*, 33(20):6494–6496, 2005.

[12] Burkhard Morgenstern Mario Stanke, Ana Tzvetkova. Augustus at egasp: using est, protein and genomic alignments for improved gene prediction in the human genome. 2006.

[13] Genı´s Parra, Pankaj Agarwal, Josep F Abril, Thomas Wiehe, James W Fickett, and Roderic Guigó. Comparative gene prediction in human and mouse. *Genome Research*, 13(1):108–117, 2003.

[14] Genis Parra, Keith Bradnam, and Ian Korf. CEGMA: a pipeline to accurately annotate core genes in eukaryotic genomes. *Bioinformatics*, 23(9):1061–1067, 2007.

[15] Søren Brunak Pierre Baldi. Bioinformatics: The machine learning approach. *Viterbi Learning*, pages 174–181, 2001.

[16] A. Krogh G. Mitchison R.Durbin, S.Eddy. Biological sequence analysis. pages 64–65, 2001.

[17] Thomas Schiex, Annick Moisan, and Pierre Rouzé. Eugene: an eukaryotic gene finder that combines several sources of evidence. In *Computational biology*, pages 111–125. Springer, 2001.

[18] M. Stanke, O. Schoffmann, B. Morgenstern, and S. Waack. Gene prediction in eukaryotes with a generalized hidden Markov model that uses hints from external sources. *BMC Bioinformatics*, 7:62, 2006.