

UNIVERZITA KOMENSKÉHO, BRATISLAVA

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

ZAROVNÁVANIE SEKVENCIÍ S POUŽITÍM METÓD
KLASIFIKÁCIE

Diplomová práca

2014

Bc. Michal Hozza

UNIVERZITA KOMENSKÉHO, BRATISLAVA

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

ZAROVNÁVANIE SEKVENCIÍ S POUŽITÍM METÓD KLASIFIKÁCIE

Diplomová práca

Študijný program: Informatika
Študijný odbor: 2508 Informatika
Školiace pracovisko: Katedra Informatiky
Školiteľ: Mgr. Tomáš Vinař, PhD.
Konzultant: Mgr. Michal Nánási

Bratislava, 2014

Bc. Michal Hozza

Podakovanie...

Bc. Michal Hozza

Abstrakt

Zarovnávanie dvoch DNA sekvencií je jedným zo základných bioinformatických problémov. Obvykle takéto zarovnanie hľadáme pomocou jednoduchých párových skrytých Markovovských modelov (pHMM). V tejto práci sa zaoberáme možnosťami použitia prídavnej informácie o funkcii vstupných sekvencií na zlepšenie kvality takýchto zarovnaní. Informácie sme zakomponovali pomocou klasifikátorov, ktoré rozhodujú či dané pozície majú byť zarovnané k sebe alebo nie. Ako klasifikátor sme použili Random forest.

Ukázalo sa, že klasifikátor sa dokáže naučiť, ktoré okná majú byť zarovnané k sebe a ktoré nie.

Vyvinuli sme 2 modely pre zarovnanie sekvencií s anotáciami za pomoci klasifikátora, ktoré sú založené na párových skrytých Markovovských modeloch.

Model s klasifikátorom ako emisiou, kde sme nahradili emisné tabuľky stavov výstupom z klasifikátora.

Model s klasifikátorovou páskou, kde navyše modelujeme aj výstup z klasifikátora.

Kľúčové slová: zarovnávanie sekvencií, strojové učenie, Random forest, anotácie

Abstract

English abstract

Key words: ...

Obsah

Úvod	1
1 Zarovnávanie sekvencií	3
1.1 Podobnosť sekvencií, sekvenčná homológia a zarovnanie	3
1.2 Párové zarovnávanie	3
1.3 Typy zarovnaní	4
1.4 Skórovacie systémy	5
1.4.1 Skórovacie matice	5
1.5 Algoritmy na hľadanie zarovnaní	5
1.5.1 Algoritmus pre globálne zarovnanie: Needleman-Wunch	6
1.5.2 Algoritmus pre lokálne zarovnanie: Smith-Waterman	7
1.5.3 Afínne skórovanie medzier	8
1.6 Zarovnávanie pomocou skrytých Markovových modelov	9
1.6.1 Skryté Markovove modely (HMM)	9
1.6.2 Viterbiho algoritmus	10
1.6.3 Nastavenie parametrov HMM	11
1.6.4 Párové zarovnávanie pomocou HMM	11
1.7 Štatistická významnosť zarovnania	11
2 Súvisiaca práca	14
3 Doplnkové informácie k sekvenciám a zdroje dát	16
4 Návrh riešenia	17
4.1 Klasifikácia na základe lokálnej informácie	17
4.1.1 Úloha klasifikátora	17

4.1.2	Random Forest	17
4.1.3	Match klasifikátor	17
4.1.4	InDel klasifikátor	18
4.2	Modely	18
4.2.1	Model s klasifikátorom ako emisiou	19
4.2.2	Model s klasifikátorovou páskou	20
5	Výsledky	21
5.1	Metódy vyhodnocovania výsledkov	21
5.2	Klasifikátor	21
5.3	Modely	21
6	Implementácia	22
6.1	Simulátor	22
6.1.1	Algoritmus	22
6.1.2	Generovanie informácie o génoch	22
6.1.3	Simulácia mutácie	23
6.1.4	Simulácia delécie	23
6.1.5	Využitie	24
	Záver	25
	Literatúra	26

Zoznam obrázkov

1.1	Lokálne zarovnanie	4
1.2	Globálne zarovnanie	6
1.3	Lokálne zarovnanie	8
1.4	Stavový diagram pre zarovnanie sekvencií	9
1.5	Párový HMM pre zarovnávanie sekvencií	12
1.6	P-hodnota lokálneho zarovnania	13
4.1	Okno klasifikátora pre pozície $i = 3$ a $j = 6$	19
4.2	Model s klasifikátorom ako emisiou	19
4.3	Model s klasifikátorovou páskou	20
6.1	Markovova reťaz použitá na generovanie informácie o génoch	23

Zoznam tabuliek

6.1 Pravdepodobnosti mutácie	23
--	----

Úvod

Najnovšie technológie sekvenovania DNA produkujú stále väčšie množstvo DNA sekvencií rôznych organizmov. Spolu s tým stúpa aj potreba rozumieť týmto dátam. Dôležitým krokom k ich porozumeniu je *zarovnávanie sekvencií* – nájdenie podobností medzi sekvenciami. Zarovnávanie sekvencií je nápomocné pri zisťovaní ich štruktúry a následne funkciu jednotlivých častí.

Existujú rôzne algoritmy na zarovnávanie sekvencií. Väčšina z nich je založená na pravdepodobnostnom modeli, pričom sa snažia nájsť zarovnanie s čo najväčšou pravdepodobnosťou. Algoritmy sú zvyčajne založené na dynamickom programovaní. Sú buď deterministické, ktoré s určitosťou nájdu najpravdepodobnejšie zarovnanie, ale za cenu kvadratického času v závislosti od dĺžok sekvencií, alebo heuristické algoritmy, ktoré nie vždy nájdu najpravdepodobnejšie zarovnanie, ale pracujú oveľa rýchlejšie.

My sa budeme zaoberať algoritmom založeným na dynamickom programovaní, kde kvalita výsledného zarovnania je ovplyvnená len pravdepodobnostným modelom.

Základný model berie do úvahy len jednotlivé *bázy* a pravdepodobnosti *substitúcie* (*mutácie*), *inzercie* a *delécie*. Naš model bude navyše uvažovať aj dodatočné informácie získané napríklad z rôznych anotátorov.

Keďže množstvo dodatočnej informácie môže byť veľmi veľké – napríklad pre 3 binárne anotátory by sme mali $2^3 \times 2^3 = 64$ krat väčší počet parametrov – nie je možné skonštruovať vhodnú skórovaciu maticu pre zarovnávací algoritmus. Namiesto nej teda použijeme klasifikátor, ktorý natrénujeme na sekvenciách so známim zarovnaním a potom použijeme na zarovnanie nových sekvencií. Klasifikátor bude vracať pravdepodobnosť, že dané dve bázy majú byť zarovnané spolu. Ako klasifikátor použijeme *Náhodný les* (*Random forest*), poprípade skúsime aj nejaké iné a porovnáme úspešnosť.

V práci sa budeme zaoberať rôznymi spôsobmi trénovania, pričom navrneme vlastné modely a porovnáme už s existujúcimi. Ďalej sa budeme zaoberať spôsobom zberu

dodatočných informácií, využitiu natrénovaného klasifikátora na zarovnávanie nových sekvencií aj pri *prezarovnaní* už existujúcich zarovnaní a implementujeme univerzálny framework, ktorý umožní využívať náš klasifikátor pri rôznych zarovnaníach.

1 Zarovnávanie sekvencií

V tejto kapitole si stručne popíšeme čo je to globálne a lokálne zarovnanie a ukážeme základné algoritmy na hľadanie globálneho a lokálneho zarovnania. Tieto algoritmy budeme neskôr modifikované používať pri našom riešení.

1.1 Podobnosť sekvencií, sekvenčná homológia a zarovnanie

V prírode vznikajú nové sekvencie modifikáciou už existujúcich (evolúciou). Preto môžeme často spozorovať podobnosť medzi neznámou sekvenciou a sekvenciou o ktorej už niečo vieme. Ak zistíme podobnosti medzi sekvenciami, môžeme preniesť informácie o štruktúre a/alebo funkcii na novú sekvenciu.

Podobné sekvencie, ktoré sa vyvinuli mutáciami so sekvencie v spoločnom predkovi sa nazývajú *homologické* a pod pojmom *hľadanie homológov* rozumieme hľadanie takých podobností, ktoré s veľkou pravdepodobnosťou vznikli práve zdieľanou evolučnou históriou.

Počas evolúcie dvoch homologických sekvencií nastane veľa *inzercií*, *delécií* a *substitúcií*, preto predtým ako môžeme začať porovnávať sekvencie, ich musíme zarovnať tak, aby homologické časti sekvencií boli na rovnakom mieste v zarovnaní. [DEKM98, BV11]

1.2 Párové zarovnávanie

Párové zarovnávanie je základná úloha zarovnávania sekvencií, kde sa k sebe zarovnávajú dve sekvencie. V tejto práci sa budeme zaoberať len párovým zarovnávaním.

Kľúčové problémy sú:

Sekvencia 1:

acgcctccacccccgcctactcgggcagtttaac
 ccttggtgttcacttgacacatcgtgaacacggcc
 cgg**CCCGACGAGAAGGCCATAATGACCTATGTGTCC**
AGCTTCTACCATGCCTTTtcaggagcgcagaaggta
 ccgagcagggccaggcaggccctcctcgccgccacc

Sekvencia 2:

tgatgccgaggatgtgttcgtcgagca
GAAGTCCATCACCTACGTGGTCACCTA
ACTTTgcaaactcaagcaggagacggt
 aagcgtatcggtaaggtaggtcggcatt
 aacgacaaaatgggtccacgactacgag

Lokálne zarovnanie:

CCCGACGAGAAGGCCATAATGACCTATGTGTCCAGCTTCTACCA-TGCCTTT
CCGGACGAGAAGTCCAT---CACCTACGTGGTCACCTACTATCACTAACTTT

Obr. 1.1: Lokálne zarovnanie

1. Aké typy zarovnávaní by sme mali uvažovať
2. Skórovací systém, ktorý použijeme na ohodnotenie zarovnaní a tréovanie
3. Algoritmus, ktorý použijeme na hľadanie optimálneho alebo dobrého zarovnaní podľa skórovacieho systému
4. Štatistická významnosť zarovnaní.

[DEKM98]

1.3 Typy zarovnaní

Základné typy zarovnaní sú *Globálne zarovnanie* a *Lokálne zarovnanie*.

Definícia 1.3.1 (Globálne zarovnanie). Vstupom sú dve sekvencie $X = x_1x_2 \dots x_n$ a $Y = y_1y_2 \dots y_m$. Výstupom je zarovnanie celých sekvencií X a Y s najvyšším skóre.

Definícia 1.3.2 (Lokálne zarovnanie). Vstupom sú dve sekvencie $X = x_1x_2 \dots x_n$ a $Y = y_1y_2 \dots y_m$. Výstupom je zarovnanie nejakých poradiacov $x_i \dots x_j$ a $y_k \dots y_l$ sekvencií s najvyšším skóre.

[BV11]

1.4 Skórovacie systémy

Takmer všetky metódy zarovnania hľadajú zarovnanie dvoch reťazcov na základe nejakej *skórovacej schémy*. Skórovacie schémy môžu byť veľmi jednoduché, napr. +1 za *zhodu* a −1 za *nezhodu*. Hoci ak chceme mať schému, kde biologicky najkorektnjšie zarovnanie má najvyššie skóre, musíme vziať do úvahy, že biologické sekvencie majú evolučnú históriu, 3D štruktúru a mnohé ďalšie vlastnosti obmedzujúce ich evolúciu. Preto skórovací systém vyžaduje starostlivé premyslenie a môže byť veľmi zložitý. [DEKM98]

1.4.1 Skórovacie matice

Skoro vždy však chceme rôzne zhody a nezhody skórovať rôzne - nie len všetky zhody +1 a nezhody −1. Skóre môže závisieť od toho aké bázy sú v danom stĺpci zarovnania. Na to môžeme použiť *skórovaciu maticu*, kde máme definované skóre pre každú dvojicu. Skórovacie matice sa využívajú najmä pri zarovnávaní proteínov, kde niektoré dvojice majú podobné chemické vlastnosti. [DEKM98, BV11]

1.5 Algoritmy na hľadanie zarovnaní

Máme daný skórovací systém, potrebujeme algoritmus, ktorý nájde optimálne zarovnanie dvoch sekvencií. Budeme uvažovať zarovnávanie s medzerami. To znamená, že môžeme do sekvencie pridať ľubovoľne veľa medzier, aby sme dosiahli lepšie skóre. Pre 2 sekvencie dĺžky n existuje

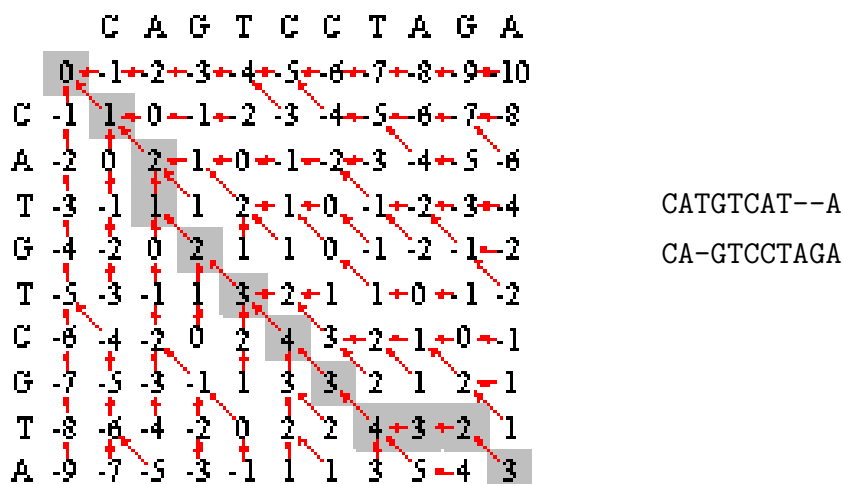
$$\binom{2n}{n} = \frac{(2n)!}{(n!)^2} \simeq \frac{2^{2n}}{\sqrt{\pi n}}$$

možných globálnych zarovnaní. Čiže nie je možné v rozumnom čase nimi prejsť.

Algoritmy na hľadanie zarovnaní využívajú *dynamické programovanie*. Algoritmy s dynamickým programovaním garantujú nájdenie optimálneho zarovnania. Existujú aj heuristické algoritmy, ktoré môžu byť veľmi rýchle, avšak majú určité predpoklady a môže sa stať, že nenájdu najlepšie zarovnanie pre niektoré páry sekvencií. My sa budeme zaoberať len algoritmami využívajúcimi dynamické programovanie. Pre rôzne typy zarovnaní máme rôzne algoritmy zarovnávaní. [DEKM98, BV11]

1.5.1 Algoritmus pre globálne zarovnanie: Needleman-Wunch

Máme dané 2 sekvencie $X = x_1x_2 \dots x_n$ a $Y = y_1y_2 \dots y_m$, budeme zarovnávať všetky znaky sekvencie X a všetky znaky sekvencie Y . Budeme používať jednoduché skórovanie: $+1$ za zhodu, -1 za nezhodu alebo pomlčku.



Obr. 1.2: Globálne zarovnanie

Algoritmus postupne vyplní 2-rozmernú maticu A . Riadky zodpovedajú bázam sekvencie X a stĺpce bázam Y . Na políčku $A[i, j]$ bude skóre najlepšieho zarovnania prvých i báz sekvencie X a prvých j báz Y .

Keď zarovnáваме sekvenciu s prázdnu sekvenciou, tak skóre bude $-n$, kde n je dĺžka sekvencie. Bude tam n pomlčiek, každá nám dá skóre -1 . Takto vyplníme riadky a stĺpce $A[i, 0]$ a $A[0, j]$.

Ak chceme vyplniť políčko $A[i, j]$, musíme si uvedomiť ako môže vyzeráť posledný stĺpec zarovnania $x_1x_2 \dots x_i$ a $y_1y_2 \dots y_j$. Máme iba 3 možnosti ako môže vyzeráť posledný stĺpec najlepšieho zarovnania. Buď obsahuje x_i alebo y_j alebo oboje. V prípade, že posledný stĺpec obsahuje oboje, cena tohto stĺpca je buď $+1$ ak $x_i = y_j$ alebo -1 ináč. Ak by sme posledný stĺpec zmazali, dostali by sme zarovnanie $x_1x_2 \dots x_{i-1}$ a $y_1y_2 \dots y_{j-1}$, pričom musí ísť o najlepšie zarovnanie. To už máme vypočítané v políčku $A[i-1, j-1]$, čiže výsledné skóre bude $A[i-1, j-1] + s(x_i, y_j)$.

V prípade, že posledný stĺpec obsahuje len x_i zarovnané s pomlčkou, skóre stĺpca bude -1 a po zmazaní dostávame zarovnanie $x_1x_2 \dots x_{i-1}$ a $y_1y_2 \dots y_j$, výsledné skóre

bude teda $A[i - 1, j] - 1$. V prípade, že posledný stĺpec obsahuje len y_i , tak skóre vypočítame analogicky.

Najlepšie skóre bude maximálne skóre pre všetky 3 prípady. Dostávame teda nasledujúci vzťah pre výpočet $A[i, j]$:

$$A[i, j] = \max \begin{cases} A[i - 1, j - 1] + s(x_i, y_j) \\ A[i - 1, j] - 1 \\ A[i, j - 1] - 1 \end{cases}$$

Maticu vieme vyplňať po riadkoch, pričom každé políčko vieme vypočítať z troch políčok, ktoré už sú vypočítané.

Ak nás zaujíma aj zarovnanie – nie len jeho skóre – vieme si pre každé políčko zapamätať ktorá z 3 možností dosiahla maximálnu hodnotu (červené šípky na Obr. 1.3). Na základe tejto informácie potom vieme zrekonštruovať zarovnanie tak, že postupne z posledného políčka ($A[n, m]$) budeme prechádzať na políčko, z ktorého sme vypočítali aktuálnu hodnotu.

Časová zložitosť je $O(nm)$, pretože vyplníme nm políčok, každé v konštantnom čase. Zjavne aj pamäťová zložitosť je $O(nm)$.

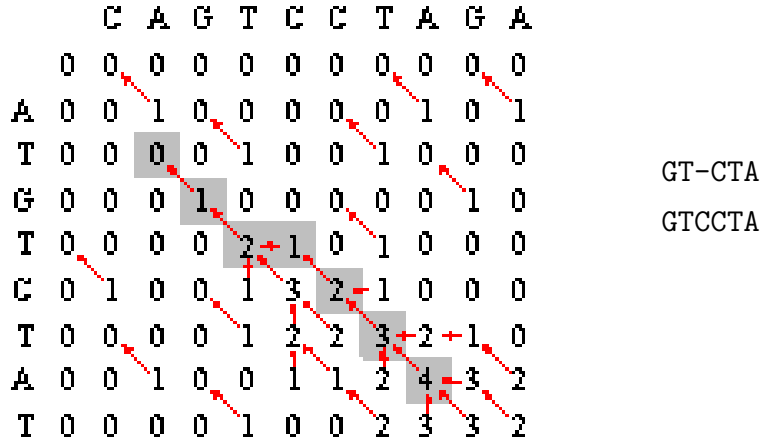
Pamäťová zložitosť sa dá zredukovať na $O(n + m)$ za cenu zhruba dvojnásobného času výpočtu [Hir75].

1.5.2 Algoritmus pre lokálne zarovnanie: Smith-Waterman

Algoritmus pre lokálne zarovnania sa líši len v niekoľkých malých detailoch. Opäť vyplníme maticu A , s tým, že v $A[i, j]$ bude najvyššie skóre lokálneho zarovnania medzi sekvenciami $x_1x_2 \dots x_i$ a $y_1y_2 \dots y_j$, ktoré buď obsahuje bázy x_i aj y_j , alebo je prázdne. Teda na ľubovoľnom mieste uvažujeme aj prázdne zarovnanie so skóre 0 (v matici nebudú záporné čísla). Vzťah pre výpočet $A[i, j]$ vyzerá takto:

$$A[i, j] = \max \begin{cases} 0 \\ A[i - 1, j - 1] + s(x_i, y_j) \\ A[i - 1, j] - 1 \\ A[i, j - 1] - 1 \end{cases}$$

Časová aj pamäťová zložitosť sú, rovnako ako pri globálnom zarovnaní $O(nm)$.



Obr. 1.3: Lokálne zarovnanie

1.5.3 Afínne skórovanie medzier

V jednoduchom skórovaní sme dávali za pomčľku vždy rovnaké skóre (-1). Pri evolúcii sa však môže stať, že sa naraz zmaže niekoľko susedných báz. Pri *afínnom skórovaní medzier* teda zavedieme dva typy skóre. Skóre za *začatie medzery* a skóre za *rozšírenie medzery*.

Algoritmus globálneho zarovnania vieme upraviť nasledovne: Namiesto matice A teraz budeme mať 3 matice M , I_x , I_y zodpovedajúce trom situáciám. **ToDo:** obrazok

Nech $M[i, j]$ je najlepšie skóre prvých i báz zo sekvencie X a prvých j báz zo sekvencie Y , pričom x_i je zarovnané k y_j , $I_x[i, j]$ je najlepšie skóre ak x_i je zarovnané k medzere a $I_y[i, j]$ je najlepšie skóre ak y_j je zarovnané k medzere.

Označme si d penaltu za začatie medzery a e penaltu za rozšírenie medzery. Vzťahy pre výpočet políček sú nasledovné:

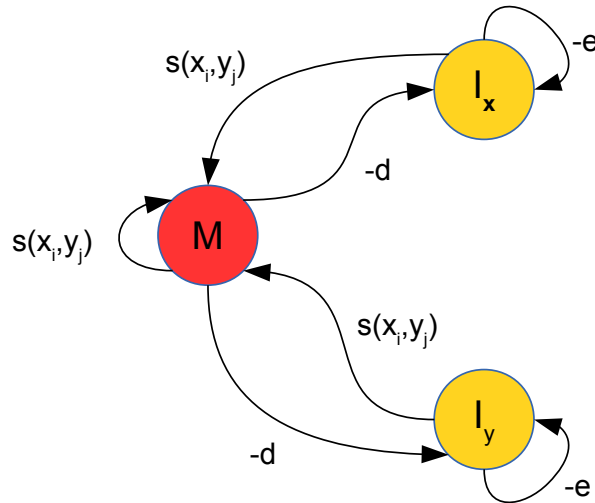
$$M[i, j] = \max \begin{cases} M[i-1, j-1] + s(x_i, y_j) \\ I_x[i-1, j-1] + s(x_i, y_j) \\ I_y[i, j-1-1] + s(x_i, y_j) \end{cases}$$

$$A[i, j] = \max \begin{cases} M[i-1, j] - d \\ I_x[i-1, j] - e \end{cases}$$

$$A[i, j] = \max \begin{cases} M[i, j-1] - d \\ I_y[i, j-1] - e \end{cases}$$

V týchto rovniciach predpokladáme, že delécia nie je nasledovaná inzerciou. Toto platí v optimálnej sekvencii, ak $-d - s$ je menšie ako najmenšie skóre nezhody.

Tieto vzťahy vieme popísať nasledovným stavovým diagramom:



Obr. 1.4: Stavový diagram pre zarovnanie sekvencií

Časová zložitosť je $O(nm)$, pretože vyplňame $3nm$ políčok, každé v konštantnom čase. Pamäťová zložitosť je opäť $O(nm)$.

[DEKM98]

1.6 Zarovnávanie pomocou skrytých Markovových modelov

1.6.1 Skryté Markovove modely (HMM)

Skrytý Markovov model (hidden Markov model, HMM) je pravdepodobnostný model, ktorý generuje náhodnú sekvenciu spolu s jej anotáciou (stavmi). HMM si môžeme predstaviť ako konečný automat. Skladá sa z niekoľkých stavov, prechodov medzi nimi a emisií. Na rozdiel od bežných konečných automatov, HMM emitujú symboly v stave, nie počas prechodu. HMM sa skladá z 3 typov pravdepodobností

- **začiatkové** pravdepodobnosť, že HMM začne v danom stave
- **prechodové** pravdepodobnosť, že HMM prejde zo stavu i do stavu j

- **emisné** pravdepodobnosť, že HMM v stave i vygeneruje symbol x

Generovanie sekvencie teda vyzerá nasledovne: Na začiatku je HMM v niektorom stave (každý stav i má nejakú pravdepodobnosť S_i , že bude začiatkový). Potom v každom kroku HMM emituje symbol x s pravdepodobnosťou $E_i[x]$ a prejde do stavu j s pravdepodobnosťou T_{ij} . Po n krokoch takto vygenerujeme sekvenciu dĺžky n , pričom každý symbol je oantovaný stavom, ktorý ho vygeneroval.

V takomto modeli vieme počítať pravdepodobnosť, že model vygeneruje danú sekvenciu dĺžky n s danou anotáciou ako súčin pravdepodobností prechodov a emisií. [BV11, DEKM98]

1.6.2 Viterbiho algoritmus

Hľadáme najpravdepodobnejšiu postupnosť stavov A , teda $\arg \max_A \Pr(A, S)$. Úlohu budeme riešiť dynamickým programovaním.

Podproblém $V[i, u]$ je pravdepodobnosť najpravdepodobnejšej cesty končiacей po i krokoch v stave u , pričom vygeneruje $S_1 S_2 \dots S_i$.

ToDo: indexy su zle, napisat to trochu inac, asi cele do toho algoritmu, a mozno tie rekurentne vzťahy zvlášť nakoniec, alebo vynechať Rekurentné vzťahy pre náš algoritmus sú nasledovné:

$$V[1, u] = \pi_u e_{u, S_1} \quad (1.1a)$$

$$V[i, u] = \max_w V[i-1, w] a_{w, u} e_{u, S_i} \quad (1.1b)$$

Algoritmus funguje takto: Nech n je dĺžka reťazca a m je počet stavov. Nainicializuj $V[1, i] \forall i$ podľa 1.1a

```

1 for i in range(2, n):
2     for u in range(1, m):
3         vypočítaj V[i, u] pomocou 1.1b

```

Maximálne $V[n, j]$ je pravdepodobnosť najpravdepodobnejšej cesty Aby sme vypísali anotáciu, pamätáme si pre každé $V[i, u]$ stav w , ktorý viedol k maximálnej hodnote vo vzorci 1.1b.

Časová zložitosť tohto algoritmu je $O(nm^2)$, kde n je dĺžka sekvencie a m počet stavov.

Poznámka: pre dlhé sekvencie budú čísla $V[i,u]$ veľmi malé a môže dôjsť k podtečeniu. V praxi teda používame zlogarimované hodnoty a namiesto násobenia súčet.

1.6.3 Nastavenie parametrov HMM

Ak máme oannotované trénovacie sekvencie, môžeme z nich parametre odvodiť frekvenčnou analýzou. Emisie získame tak, že vyfiltrujeme symboly s príslušným stavom a spočítame výskyty pre každý stav zvlášť a tranzície získame tak, že pre každý stav spočítame frekvencie nasledujúcich stavov.

1.6.4 Párové zarovnávanie pomocou HMM

V časti 1.5.3 sme si ukázali jednoduchý algoritmus na globálne zarovnávanie s afínnym skórovaním medzier. K tomuto algoritmu sme si uviedli aj jednoduchý stavový automat (Obr. 1.4). Tento automat vieme previesť na HMM.

Na to aby sme automat previedli na HMM, musíme urobiť niekoľko zmien - musíme nastaviť emisné a tranzičné pravdepodobnosti, tak aby sumovali do jedna. Pre jednoduchosť pridáme aj prechody medzi stavmi XaY . Ak ich pravdepodobnosti nastavíme na 0, máme model ekvivalentný predchádzajúcemu.

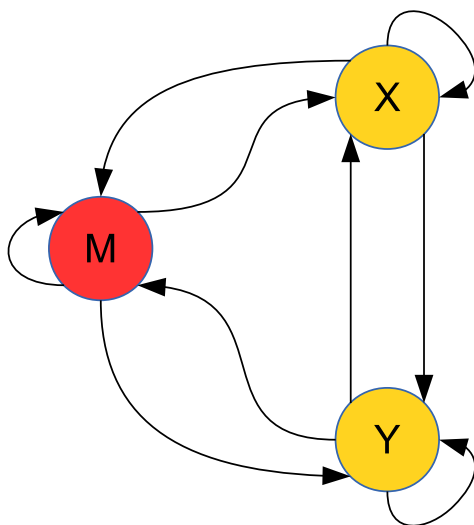
Dostaneme model podobný HMM s tým rozdielom, že namiesto jedného symbolu emitujeme dvojicu symbolov. Takýto model sa nazýva *Párový skrytý Markov model*. Na tento model môžeme použiť mierne modifikovaný Viterbiho algoritmus na nájdenie najpravdepodobnejšej postupnosti stavov, čo nám dá najpravdepodobnejšie zarovnanie.

Parametre modelu môžeme ľahko natrénovať z existujúcich párových zarovnaní.

Nech dĺžky oboch sekvencií sú $O(n)$, a m je počet stavov. Potom Viterbiho algoritmus na Párovom HMM bude bežať v čase $O(n^2m^2)$. [DEKM98]

1.7 Štatistická významnosť zarovnania

Smith-Watermanov algoritmus nájde najlepšie lokálne zarovnanie, pre ľubovoľné dve sekvencie. Treba však rozhodnúť, či je zarovnanie dostatočne vierohodné na to, aby



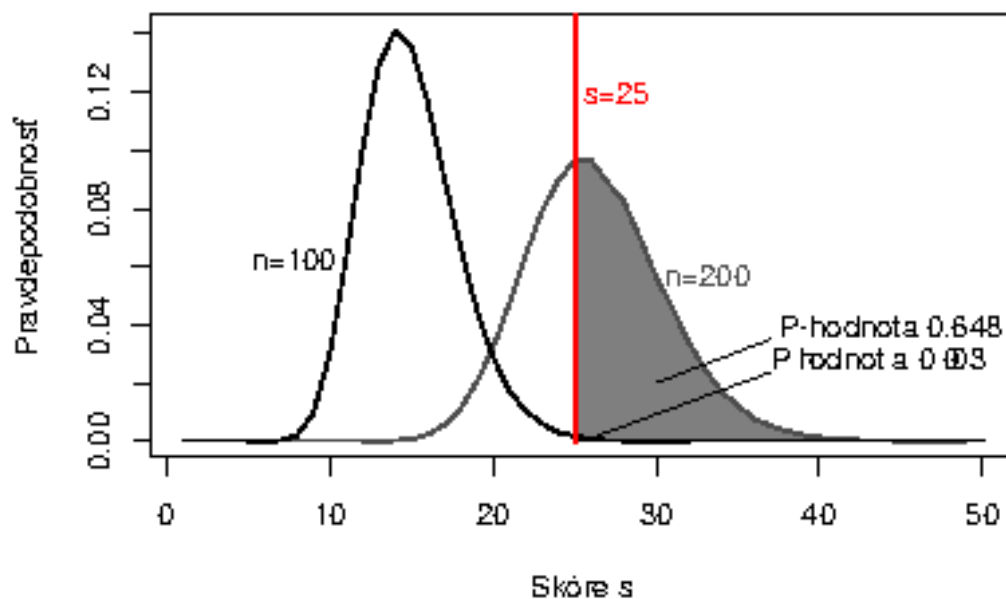
Obr. 1.5: Párový HMM pre zarovnávanie sekvencií

predstavovalo skutočnú podobnosť sekvencií a nie len najlepšie zarovnanie dvoch nesúvisiacich sekvencií. Ako vodítko pri rozhodnutí sa používajú identifikátory *štatistickej významnosti* zarovnania: *P-hodnota* (*P-value*) alebo *E-hodnota* (*E-value*).

P-hodnota zarovnania je pravdepodobnosť, že medzi náhodne generovanými sekvenciami tej istej dĺžky by sme našli zarovnanie s rovnakým skóre alebo vyšším. Keďže P-hodnota závisí od dĺžok sekvencií a skóre, musíme ju počítať pri každom zarovnaní. Je však časovo náročné robiť to generovaním veľkého množstva zarovnaní, preto sa používajú matematicky odvodené vzorce na odhad tejto hodnoty. ([KA90], [MB06]).

E-hodnota vyjadruje strednú hodnotu počtu zarovnaní so skóre aspoň takým ako má naše zarovnanie medzi náhodne generovanými sekvenciami. E-hodnota teda môže byť aj väčšia ako jedna. Ak je E-hodnota väčšia ako jedna, tak čisto náhodou by sme očakávali aspoň 1 také silné zarovnanie a teda zarovnania s tak vysokou E-hodnotou nebudeme považovať za štatisticky významné.

V štatistike sa v rôznych testoch štandardne používajú prahy na P-hodnotu 0.05 alebo 0.01. Pri zarovnávaní sekvencií však často používame ešte nižší prah, teda uvažujeme len zarovnania s P-hodnotou menšou ako napr. 10^{-5} . Pri malých hodnotách sú P-hodnota a E-hodnota preblížne rovnaké, teda taký istý prah môžeme použiť aj na E-hodnotu.



Obr. 1.6: P-hodnota lokálneho zarovnania

P-hodnota lokálneho zarovnania so skóre $s = 25$ medzi 2 sekvenciami dĺžky $n = 100$ alebo $n = 200$ (skórovanie $+1$ zhoda, -1 nezhoda alebo medzera). Rozdelenie bolo získané zarovnávaním 100000 párov náhodných sekvencií. Pri $n = 100$ je P-hodnota približne 0.003 a zodpovedá malej čiernej poloche pod krivkou napravo od zvislej čiary pre $s = 25$. Pri dlhších sekvenciách zodpovedá P-hodnota veľkej sivej ploche napravo od zvislej čiary. Pri takto dlhých sekvenciách očakávame skóre 25 alebo väčšie vo viac ako 60% prípadov čisto náhodou. Nejde teda o štatisticky významné zarovnanie.

2 Súvisiaca práca

V tejto kapitole si uvedieme stručný prehľad modelov, ktoré zahŕňajú doplnkové informácie do zarovnania pomocou metód klasifikácie a stručne uvedieme v čom sa bude náš model líšiť.

V princípe môžeme rozlišovať dva typy modelov - *generatívny model* a *diskriminačný model*.

Konvenčné techniky odhadu pre zarovnávanie sa zakladajú na generatívnom modeli. Generatívny model (napr. HMM) sa snaží modelovať proces, ktorý generuje dáta ako pravdepodobnosť $P(X, Y, Z)$, kde $X = x_1x_2 \dots x_n$, $Y = y_1y_2 \dots y_m$ a Z je zarovnanie. Ak poznáme $P(X, Y, Z)$ (alebo jej dobrý odhad),

$$\arg \max_z P(X = x, Y = y, Z = z)$$

predikuje zarovnanie z z dvoch sekvencií x a y . Aby sme žľahčili odhad $P(X, Y, Z)$, rozložíme ju pomocou nezávislých predpokladov na procese, ktorý generuje x a y . To síce vedie k efektívnym a jednoduchým problémom odhadu, ale obmedzuje to interakcie v rámci sekvencií, ktoré by sme mohli modelovať. [YJEP07]

Výskum v oblasti strojového učenia dokázal, že diskriminatívne učenie (SVM, RandomForest) zvyčajne produkuje oveľa presnejšie pravidlá ako generatívne učenie (HMM, naive Bayes classifier). Môže to byť vysvetlené tým, že $P(Z|X, Y)$, je už vhodné na vyhodnotenie optimálnej predikcie

$$\arg \max_z P(Z = z | X = x, Y = y).$$

Diskriminačné učenie aplikované na problém zarovnania bude priamo odhadovať $P(Z|X, Y)$ alebo prislúchajúcu diskriminačnú funkciu, a preto sa zamerá na podstatnú časť problému odhadu. [YJEP07]

Aktuálne existuje len niekoľko prístupov k diskriminačnému učeniu modelov zarovnávaní. Jeden z možných prístupov je riešiť *problém inverzného zarovnania* pomocou

strojového učenia. [YJEP07]

Definícia 2.0.1 (Inverzné zarovnanie). Máme dané sekvencie a k nim zarovnanie. Inverzné zarovnanie nám vráti váhový model, s ktorým daný algoritmus na zarovnávanie vráti požadované zarovnanie k daným sekvenciám.

Problém inverzného zarovnania bol prvý krát formulovaný v [GS96]. Na tomto probléme je postavený aj model v [YJEP07], kde sa na tréovanie Support Vector Machine (SVM) dá pozerat ako na riešenie tohto problému. V článku sa zaoberajú použitím *Structural SVM* algoritmu na zarovnávanie proteínových sekvencií. Diskriminatívne učenie umožňuje zahrnutie množstva dodatočnej informácie – státisíce parametrov. Navyše SVM umožňuje tréovanie pomocou rôznych účelových funkcií (loss functions). SVM algoritmus má lepšiu úspešnosť ako generatívna metóda SSALN, ktorá je veľmi presným generatívnym modelom zarovnaní, ktorá zahŕňa informáciu o štruktúre.

Podobný prístup je aj v CONTRAlign [DGB06], kde sa používajú Conditional Random Fields (CRF). Tento prístup tiež ťaží z benefitov diskriminatívneho učenia, avšak narozdiel od [YJEP07] neumožňuje použitie účelových funkcií závislých na aplikácii.

3 Doplnkové informácie k sekvenciám a zdroje dát

4 Návrh riešenia

4.1 Klasifikácia na základe lokálnej informácie

4.1.1 Úloha klasifikátora

V našej práci sme klasifikátor použili na zakomponovaní dodatočných informácií o sekvenciách do modelu zarovnania sekvencií. Dodatočné informácie sú poskytnuté formou anotácií k príslušným bázam.

V našich modeloch sme použili 2 typy klasifikátorov – *Match klasifikátor* a *InDel klasifikátor*. Match klasifikátor sa klasifikátor určuje s akou pravdepodobnosťou sa majú dané dve pozície v sekvenciách zarovnať k sebe. InDel klasifikátor určuje s akou pravdepodobnosťou má byť pozícia v príslušnej sekvencii zarovnaná s medzerou. Tieto pravdepodobnosti sú mierou istoty daného klasifikátora, a keďže tieto 2 klasifikátory sú nezávislé, súčet ich výstupov nemusí byť jedna.

4.1.2 Random Forest

ToDo: chcem to tu, alebo niekde inde?

ToDo:

4.1.3 Match klasifikátor

ToDo: možno to nechcem striktne rozdeľovať na tieto 2 klasifikátory, ak budem mať rôzne typy dát tak to možno budem chcieť radšej zlúčiť do jedného

Match klasifikátor určuje, či sa dané dve pozície majú zarovnať k sebe. Jeho výstupom je číslo z intervalu $\langle 0, 1 \rangle$, pričom čím bližšie je toto číslo k 1, tým si je klasifikátor istejší, že dané 2 pozície sa majú zarovnať k sebe. Naopak, čím bližšie je k 0, tým si je viac istý, že by tieto pozície k sebe byť nemajú.

Vstupné dáta

Ako vstupné dáta dostane tento klasifikátor okolie okolo daných pozícií. Toto okolie budeme volať *okno*.

Majme dve sekvencie, $X = x_1x_2 \dots x_n$ a $Y = y_1y_2 \dots y_n$ a pozície i a j . Okno veľkosti w obsahuje $x_{i-w/2} \dots x_i \dots x_{i+(1+w)/2}$, $y_{j-w/2} \dots y_j \dots y_{j+(1+w)/2}$ a všetky anotácie príslušných báz. (Obr. 4.1a)

Trénovanie

ToDo:

4.1.4 InDel klasifikátor

InDel klasifikátor určuje, či sa daná pozícia má zarovnať k medzere. Jeho výstupom je opäť číslo z intervalu $\langle 0, 1 \rangle$, pričom čím bližšie je toto číslo k 1, tým si je klasifikátor istejší, že daná pozícia sa má zarovnať k medzere a čím bližšie je k 0, tým si je viac istý, že sa táto pozícia nemá zarovnať k medzere.

Vstupné dáta

Aj pri tomto klasifikátore používame dve pozície prvá je pozícia v inzert sekvencii a ukazuje na bázu, na ktorú sa pýtame a druhá pozícia je v druhej sekvencii a ukazuje na medzeru - teda medzi dve bázy.

Majme dve sekvencie, $X = x_1x_2 \dots x_n$ a $Y = y_1y_2 \dots y_n$ a pozície i a j . Nech X je inzert sekvencie, potom okno veľkosti w obsahuje $x_{i-w/2} \dots x_i \dots x_{i+(1+w)/2}$, $y_{j-w/2} \dots y_j \dots y_{j+(1+w)/2-1}$ a všetky anotácie príslušných báz. (Obr. 4.1b)

Trénovanie

ToDo:

4.2 Modely

V sekcii 1.6 sme si zadefinovali HMM pre zarovnávanie sekvenci (obr. 1.5). V našom riešení sme predstavili 2 modifikácie pôvodného HMM na zakomponovanie dodatočnej

$i:012 \quad 34567$
 $Ax:000 \quad 11111$
 $X:ACG-----TGTTC$
 $Y:ACCATTCCTA--C$
 $Ay:000111111 \quad 0$
 $j:012345678 \quad 9$

(a) Match klasifikátor

$i:012 \quad 34567$
 $Ax:000 \quad 11111$
 $X:ACG-----TGTTC$
 $Y:ACCATTCCTA--C$
 $Ay:000111111 \quad 0$
 $j:012345678 \quad 9$

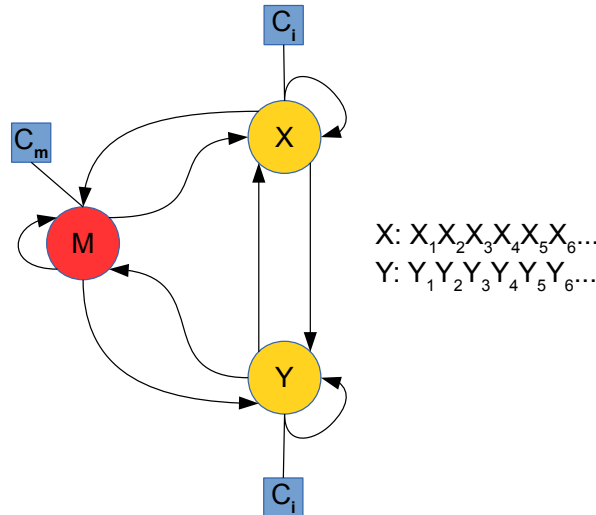
(b) InDel klasifikátor

Obr. 4.1: Okno klasifikátora pre pozície $i = 3$ a $j = 6$

informácie, pričom sme využili klasifikátory. V oboch modeloch sú klasifikátory rovnaké, aj s rovnakým postupom tréovania. Líši sa len tréovanie samotného HMM a architektúra modelu.

4.2.1 Model s klasifikátorom ako emisiou

V tomto modeli sme nahradili emisné tabuľky stavov výstupom z klasifikátora. Model teda bude vyzeráť rovnako, aj pravdepodobnosti prechodov zostanú, ale emisná pravdepodobnosť sa nahradí výstupom z klasifikátora.



Obr. 4.2: Model s klasifikátorom ako emisiou

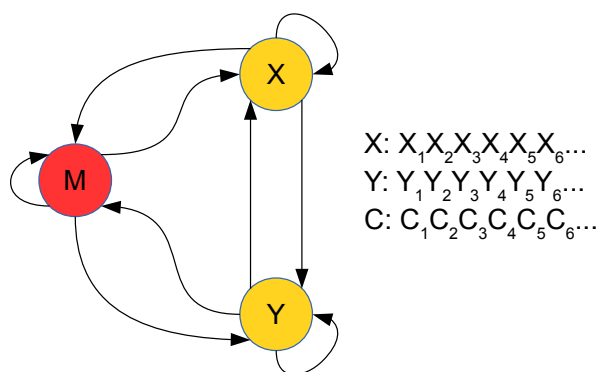
Problémom tohto modelu je, že výstup klasifikátora nezodpovedá emisným pravde-

podobnostiam, ale akejsi istote klasifikátora o tom, že dve pozície majú byť zarovnané k sebe. Hodnoty z klasifikátora teda nesumujú do 1 a model nie je celkom korektný. V praxi sa však ukázalo, že to až tak nevádi, avšak o takomto modeli už nemôžeme hovoriť ako o pravdepodobnostnom. Je len inšpirovaný HMM.¹

V tomto modeli sme trénovali iba tranzície, emisie sme mali priamo z natrénovaného klasifikátora.

4.2.2 Model s klasifikátorovou páskou

Na to aby sme vyriešili problém s korektnosťou predošlého modelu, navrhli sme alternatívny model, ktorý navyše modeluje aj výstup z klasifikátora. Nemodelujeme teda len dvojicu sekvencií, ale aj sekvenciu výstupov klasifikátora. Pásku s výstupom z klasifikátora vieme považovať za akýsi hint pre náš zarovnávač.



Obr. 4.3: Model s klasifikátorovou páskou

Tento model je síce narozdiel od predošlého korektný pravdepodobnostný model, no má však jednu nevýhodu. Že okrem prípadov, keď klasifikátor vráti hodnotu blízku 0 – teda tvrdí, že dané 2 pozície by nemali byť zarovnané k sebe (resp. daná pozícia by nemala byť zarovnaná k medzere), penalizuje aj prípady kedy klasifikátor vracia hodnoty blízke 1. Je to z dôvodu, že HMM sa trénuje pomocou frekvenčnej analýzy a pozícií, kde sa vyskytujú hodnoty blízke 1 je menej ako pozícií s hodnotami okolo 0.7.

V tomto modeli sme trénovali aj tranzície aj emisie. Klasifikátorovú pásku sme generovali pomocou oboch klasifikátorov, pričom v Match stave sme použili Match klasifikátor a v Insert stavoch InDel klasifikátor. Výstupy z klasifikátora sme rozdelili do 10 košov rovnomerne na intervale $\langle 0, 1 \rangle$

5 Výsledky

5.1 Metódy vyhodnocovania výsledkov

ToDo: klasifikator

ToDo: zhoda

ToDo: traniztivita

5.2 Klasifikátor

5.3 Modely

6 Implementácia

6.1 Simulátor

Simulátor slúži na overenie funkčnosti zarovnávača. Náhodne vygeneruje 2 sekvencie, ktoré vznikli zo spoločného predka a vyrobí korektné zarovnanie. Okrem toho vyrobí aj nejaké dodatočné informácie ktoré majú pomôcť pri zarovnávaní.

6.1.1 Algoritmus

Simulátor vygeneruje informáciu o tom, ktoré časti sekvencie prislúchajú génom a ktoré nie. Informácia má podobu boolovskeho vektora. Simulátor najskôr vygeneruje *základnú (master) postupnosť* a z nej odvodí dve ďalšie postupnosti, ktoré zodpovedajú sekvenciám.

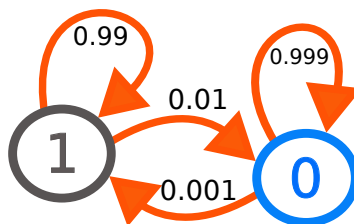
Okrem toho simulátor vygeneruje dve sekvencie, pričom prvú vyrobí náhodne a druhú odvodí z nej pomocou mutácie a inzercie/delécie. V našom prípade sme inzerciu vynechali a simulujeme ju ako deléciu v druhej sekvencii. Pri odvodzovaní bude používať aj informáciu o tom, ktorá časť je gén a ktorá nie.

Keďže simulátor vie spôsob akým generoval druhú sekvenciu z prvej, vie aj korektné zarovnanie.

Simulátor má vopred daných niekoľko konštánt – pravdepodobnosti udalostí, ktoré môžu nastať.

6.1.2 Generovanie informácie o génoch

Ak sa na danom mieste nachádza gén, označíme to 1 inak 0. Gény bývajú súvislé úseky, takže ich treba generovať tak, že niekedy začneme gén, potom generujeme 1, potom skončíme gén a generujeme 0. Potom môžeme opäť začať gén atď.



Obr. 6.1: Markovova reťaz použitá na generovanie informácie o génoch

Generovanie robíme pomocou *Markovovej reťaze* (*Markov Chain*) obr. 6.1. Generujeme podľa aktuálneho stavu a v každom kroku sa podľa pravdepodobnosti rozhodneme či sa prepne do iného stavu alebo ostaneme v tom istom. Rozhodnutie robíme pomocou *falošnej mince* (*biased coin*), kde hlava padne s určitou pravdepodobnosťou, ktorú vopred nastavíme.

Máme vygenerovanú master postupnosť a z nej teraz vyrobíme dve postupnosti pre sekvencie tak, že skopírujeme master sekvenciu, pričom každú 1 s určitou pravdepodobnosťou (v našom prípade 0,1) zmeníme na 0.

6.1.3 Simulácia mutácie

Máme vygenerovanú sekvenciu a ideme vyrobiť zmutovanú sekvenciu. Spravíme to tak, že s určitou pravdepodobnosťou sa nahradí báza z pôvodnej sekvencie inou bázou. Pravdepodobnosť závisí aj od toho, či je na danej pozícii gén v oboch sekvenciách, v jednej, alebo v žiadnej. Na rozhodnutie používame jednu z troch falošných mincí podľa toho, ktorá z možností nastala (tabuľka 6.1).

Gén A	0	0	1	1
Gén B	0	1	0	1
Pravdepodobnosť	0,35	0,3	0,3	0,2

Tabuľka 6.1: Pravdepodobnosti mutácie

6.1.4 Simulácia delécie

Deléciu simulujeme opäť pomocou Markovovej reťaze, pretože pri evolúcii majú tendenciu vypadávať súvislé úseky. Pravdepodobnosť, že začneme mazať je 0,01 a že

prestaneme 0, 1. Ak mažeme, nahradzujeme danú bázu znakom '–'.

6.1.5 Využitie

Simulátor je prvá vec, ktorú sme implementovali a slúžil hlavne na prvotné experimenty.

ToDo: vieme ho prispodobovať a merať na nom korektnosť a užitočnosť modelu, alebo niečo na ten štýl

Záver

Literatúra

- [BC] Leo Breiman and Adele Cutler. Random forests. [Online; accessed 14-Jan-2013].
- [BPSS11] Alvis Brazma, Helen Parkinson, Thomas Schlitt, and Mohammadreza Shojatalab. EBI Research - Functional Genomics - Introduction To Biology. 2011. [Online; accessed 26-Oct-2012].
- [Bre01] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- [BV11] Broňa Brejová and Tomáš Vinař. *Metódy v bioinformatike [Methods in Bioinformatics]*. Knižničné a edičné centrum FMFI UK, 2011. Lecture notes.
- [DEKM98] R. Durbin, S.R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.
- [DGB06] Chuong Do, Samuel Gross, and Serafim Batzoglou. Contralign: Discriminative training for protein sequence alignment. In Alberto Apostolico, Concettina Guerra, Sorin Istrail, Pavel Pevzner, and Michael Waterman, editors, *Research in Computational Molecular Biology*, volume 3909 of *Lecture Notes in Computer Science*, pages 160–174. Springer Berlin / Heidelberg, 2006. 10.1007/11732990_15.
- [GS96] D. Gusfield and P. Stelling. [28] parametric and inverse-parametric sequence alignment with xparal. *Methods in enzymology*, 266:481–494, 1996.
- [Hir75] D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Commun. ACM*, 18(6):341–343, June 1975.

- [KA90] S Karlin and S F Altschul. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proceedings of the National Academy of Sciences*, 87(6):2264–2268, 1990.
- [MB06] Alexander Yu. Mitrophanov and Mark Borodovsky. Statistical significance in biological sequence analysis. *Briefings in Bioinformatics*, pages 2–24, 2006.
- [NW70] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443 – 453, 1970.
- [Sri] Sargur N. Srihari. Machine Learning: Generative and Discriminative Models.
<http://www.cedar.buffalo.edu/~srihari/CSE574/Discriminative-Generative.pdf>. [Online; accessed 14-Jan-2013].
- [Sut07] Ivan Sutóris. Tvorba klasifikačných stromov pri procese data miningu, 2007.
- [SW81] T.F. Smith and M.S. Waterman. Identification of common molecular sub-sequences. *Journal of Molecular Biology*, 147(1):195 – 197, 1981.
- [YJEP07] Chun-Nam Yu, Thorsten Joachims, Ron Elber, and Jaroslaw Pillardy. Support vector training of protein alignment models. In Terry Speed and Haiyan Huang, editors, *Research in Computational Molecular Biology*, volume 4453 of *Lecture Notes in Computer Science*, pages 253–267. Springer Berlin / Heidelberg, 2007. 10.1007/978-3-540-71681-5_18.