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

Poďakovanie...

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

Zoznam obrázkov

Zoznam tabuliek

Úvod

Najnovšie technológie sekvenovania DNA produkujú stále väčšie množstvo sekvencií rôznych organizmov. Spolu s tým stúpa aj potreba rozumieť týmto dátam. Dôležitým krokom k ich porozumeniu je *zarovnávanie sekvencií.* Zarovnávanie dvoch DNA sekvencií je teda jedným zo základných bioinformatických problémov. Správne zarovnanie identifikuje časti sekvencie, ktoré vznikli z toho istého predka (zarovnané bázy), ako aj inzercie a delécie v priebehu evolúcie (medzery v zarovnaní). 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 a pracujú v kvadratickom čase v závislosti od dĺžok sekvencií. Niekedy sa na urýchlenie použijú rôzne heuristické algoritmy, ktoré nie vždy nájdu najpravdepodobnejšie zarovnanie, ale pracujú oveľa rýchlejšie.

My sme sa v práci zaoberali algoritmom, ktorý hľadá zarovnanie pomocou jednoduchých párových skrytých Markovovských modelov (pHMM) [DEKM98], kde kvalita výsledného zarovnania je ovplyvnená len pravdepodobnostným modelom.

Základný model berie do úvahy len jednotlivé *bázy* a pravdepodobnosti *substitúcie (mutácie)*, *inzercie* a *delécie.* Náš model navyše uvažuje aj prídavné informácie (takzvané anotácie) získané z externých programov (napr. anotácie o génoch z vyhľadávača génov).

Keďže množstvo dodatočnej informácie môže byť veľmi veľké – napríklad pre 3 binárne anotácie by sme mali $2^3 \times 2^3 = 64$ krát väčší počet parametrov – je ťažké skonštruovať vhodnú skórovaciu maticu pre zarovnávací algoritmus. Namiesto nej sme teda použili klasifikátory[1], ktorý sme trénovali na sekvenciách so známym zarovna-

[1]program, ktorý na základe vstupnej informácie a vopred natrénovaných parametrov klasifikuje dáta

ním a potom použili na zarovnanie nových sekvencií. Naše klasifikátory vracajú čísla z intervalu $\langle 0, 1\rangle$, ktoré určujú, či dané dve bázy majú byť zarovnané spolu.

Ako klasifikátor sme použili *RandomForest* [Bre01], pretože aktuálne patrí medzi najlepšie klasifikátory.

ToDo: napisat este strucny obsah prace

do niektorej triedy z danej množiny tried

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 evolúciou nové sekvencie modifikáciou už existujúcich. Preto môžme 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:

```
gagacccgcctaggtgaatatttagcagc
gattaaataccacgtaTATAAGGTGGACC
GTTCCTCGAGAGGTTCTTCCGGCAATGAC
GGCCAGAGCAAAAGCCACGTgtaggactg
catacgcctctacgcctccactgacgcga
tgatgtggcgtggatctgtttgctcttgg
tataggtcacggagacggctggtactgat
cccttcgggagtaaaaatataatgaccat
ggcccaggcttcaggagggagttgtgcgg
```

Sekvencia 2:

```
tgtacagcactgcaacgagcatctggggg
ttggttattccgatggcgctggacagcta
gcggacagtagttctcaggccttagtaga
aaggtgggaaccccccTATGAGGTCGACCG
TTTCAGCGTGACTATAGACGTCATTGAAG
CAATATACAGGAACACCACCTacttagga
agggagttcggtgcagtaaagcattctta
cctcagggcacggtagagaacactacaac
cagaatagcaacgtgatgcggcgactctc
```

Lokálne zarovnanie:

```
TATAAGGTGGACCGTT--------CCTCGAGAGGTTCTTCCGGCAATGGCCACGAGAGCAAAAGCCACGT
TATGAGGTCGACCGTTTCAGCGTGACTATAGACGTCATTGAAGCAATATACAGG------AACACCACCT
```

Obr. 1.1: Dve sekvencie a ich lokálne zarovnanie. Veľkými písmenami a červenou farbou sú vyznačené zarovnané časti. V zarovnaní sa nachádzajú zhody, nezhody a medzery v oboch sekvenciách

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

[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.

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 podreťazcov $x_i \dots x_j$ a $y_k \dots y_l$ sekvencií.

V praxi sa väčšinou snažíme nájsť zarovnania s najvyšším skóre. [BV11]

1.4 Skórovacie systémy

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

[1]metóda, ktorou priradíme zarovnaniu skóre - zvyčajne čím väčšie skóre, tým realistickejšie zarovnanie by to malo byť

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 sa používa *skórovacia matica* (obr. 1.2), kde máme definované skóre pre každú dvojicu. Skórovacie matice sa využívajú najmä pri zarovnávaní proteínov, kde niektoré dvojice majú podobné chemické vlastnosti. [DEKM98, BV11]

Iným typom skórovacej schémy je napríklad *párový skrytý markvovský model* (viac v 1.6)

	A	C	G	T	-
A	6	-1	-2	-1	-3
C	-1	5	-3	-2	-4
G	-2	-3	5	-2	-2
T	-1	-2	-2	6	-1
-	-3	-4	-2	-1	∞

Obr. 1.2: Ukážka skórovacej matice. Všimnime si, že môžu byť rôzne skóre aj za jednotlivé zhody, nezhody alebo medzery.

1.5 Algoritmy na hľadanie zarovnaní

Pre danú skórovaciu schému potrebujeme algoritmus, ktorý nájde optimálne zarovnanie dvoch sekvencií. Budeme uvažovať zarovnávanie s medzerami. Medzery používame na znázornenie delécie v danej sekvencii, alebo inzercie v druhej sekvencii. Na označenie medzier používame pomlčku `'-'` a medzeru do sekvencie medzi 2 bázy pridáme tak, že medzi 2 bázy napíšeme jednu alebo viac pomlčiek - počet pomlčiek zodpovedá poctu medzier a teda aj počtu báz, ktoré má na danom mieste jedna sekvencia navyše oproti druhej. Do sekvencie môžeme pridať ľubovoľne veľa medzier, aby sme dosiahli lepšie skóre. Pre 2 sekvencie dĺžky n existuje

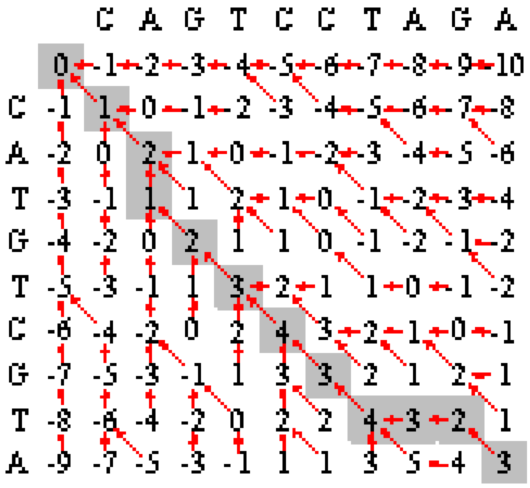
$$\binom{2n}{n} = \frac{(2n)!}{(n!)^2} \simeq \frac{2^{2n}}{\sqrt{\pi n}}$$

možných globálnych zarovnaní. [DEKM98] Čiže nie je možné v rozumnom čase nimi prejsť.

Algoritmy na hľadanie zarovnaní využívajú *dynamické programovanie*. Často sa používajú aj rôzne heuristiky na urýchlenie výpočtu. My sa budeme zaoberať len algoritmami využívajúcimi dynamické programovanie. Pre rôzne typy zarovnaní a skórovacie schémy máme rôzne algoritmy zarovnávania. [DEKM98, BV11]

1.5.1 Algoritmus pre globálne zarovnanie: Needelman-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. Definujeme si jednoduchú skórovaciu tabuľku kde $s(x, y)$ bude udávať skóre pre danú dvojicu báz (napr. $+1$ za zhodu, -1 za nezhodu) a za nezhodu budeme dávať penaltu $-d$.




```
CATGTCAT--A
CA-GTCCTAGA
```

Obr. 1.3: Tabuľka dynamického programovania pre globálne zarovnanie (vľavo) a výsledné zarovnanie (vpravo). Skóre je +1 za zhodu a -1 za nezhodu alebo medzeru.

Algoritmus postupne vypĺňa 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 zarovnávame sekvenciu s prázdnou 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 vyzerať posledný stĺpec zarovnania $x_1x_2 \dots x_i$ a $y_1y_2 \dots y_j$. Máme iba 3 možnosti ako môže vyzerať posledný stĺpec najlepšieho zarovnania. Buď obsahuje x_i alebo y_j alebo oboje. V prípade, že posledný stĺpec obsahuje oboje, cena tohto stĺpca je $s(x_i, y_j)$. 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 $A[i, j]$:

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

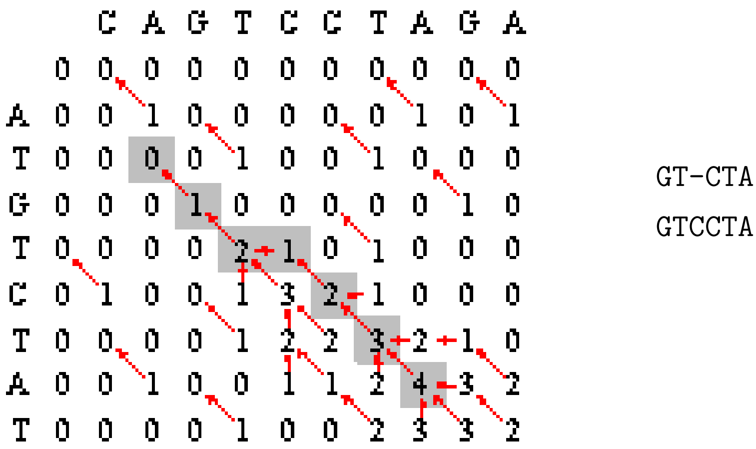
Maticu vieme vypĺňať po riadkoch, pričom každé políčok vieme vypočítať z troch políčok, ktoré už sú vypočítané. Políčko $A[0, 0] = 0$ a krajné políčka potom vieme vypočítať ako $A[i, 0] = A[i-1, j] - d$ a $A[0, j] = A[i, j-1] - d$

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 vypĺňame 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



Obr. 1.4: Tabuľka dynamického programovania pre lokálne zarovnanie (vľavo) a výsledné zarovnanie (vpravo). Skóre je +1 za zhodu a -1 za nezhodu alebo medzeru.

Algoritmus pre lokálne zarovnania sa líši len v niekoľkých malých detailoch. Opäť vypĺňame 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:

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

V tomto prípade sú všetky krajné políčka nulové.

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

1.5.3 Afínne skórovanie medzier

V jednoduchom skórovaní sme dávali za pomlčku 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 (Obr. 1.5).

(a) Mutácia(M)	(b) Inzercia v X (I_x)	(c) Inzercia v Y (I_y)
$\mathtt{ACTx_i}$ $\mathtt{AGTy_j}$	$\mathtt{ACTTAx_i}$ $\mathtt{AGTy_j{-}{-}}$	$\mathtt{ACTx_i{-}{-}}$ $\mathtt{AGTATy_j}$

Obr. 1.5: Tri situácie pri afínnoom skórovaní medzier

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 políčok sú nasledovné:

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

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

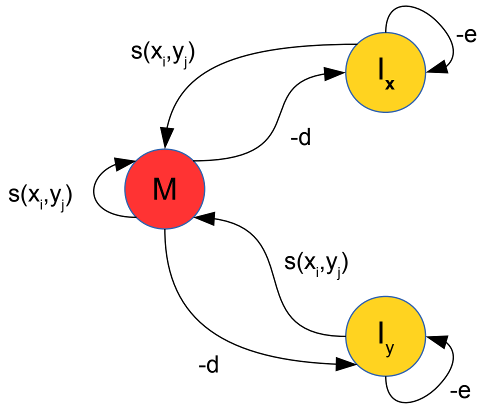
$$A[i,j] = \max\begin{cases} M[i,j-1] - d \\ I_y[i,j-1] - e \end{cases}$$

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 stavovým diagramom na obrázku 1.6:

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

[DEKM98]



Obr. 1.6: Stavový diagram pre zarovnanie sekvencií - obsahuje *Match* (M), *InsertX* (I_x) a *InsertY* (I_y) stav a prechody medzi nimi spolu s ich cenou. Napríklad prechod z M do I_x znamená vloženie medzery do Y-ovej sekvencie a to penalizujeme $-d$

1.6 Zarovnávanie pomocou skrytých Markovovských modelov

1.6.1 Skryté Markovovské modely (HMM)

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

- distribúcia začiatočných stavov (HMM začne v stave i)
- distribúcia prechodov (HMM prejde zo stavu i do stavu j)
- distribúcia emisií (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 π_i, že bude začiatočný). Potom v každom kroku HMM emituje symbol x s pravdepodobnosťou $e_{i,x}$ a prejde do stavu j s pravdepodobnosťou $a_{i,j}$. Po n krokoch takto vygenerujeme sekvenciu dĺžky n, pričom každý symbol je oanotovaný stavom, ktorý ho vygeneroval.

V takomto modeli vieme počítať pravdepodobnosť, že model vygeneruje sekvenciu x dĺžky n s anotáciou s ako súčin pravdepodobností prechodov a emisií. Výpočet vyzerá nasledovne:

$$P[X = x|S = s] = \pi_{s_1} e_{s_1,x1} a_{s_1,s_2} e_{s_2,x_2} a_{s_2,s_3} e_{s_3,x_3} \dots a_{s_{n-1},s_n} e_{s_n,x_n}$$

. [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 končiacej po i krokoch v stave u, pričom vygeneruje $s_1 s_2 \dots s_i$.

Rekurentné vzťahy pre náš algoritmus sú nasledovné:

$$V[1, u] = \pi_u e_{s_1,u} \tag{1.1a}$$

$$V[i, u] = \max_w V[i-1, w] a_{w,u} e_{s_i,u} \tag{1.1b}$$

Algoritmus funguje takto: Nech n je dĺžka reťazca a m je počet stavov.

```
Nainicializuj V[1,i] ∀i podľa 1.1a
for i in range(2, n):
    for u in range(1, m):
        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

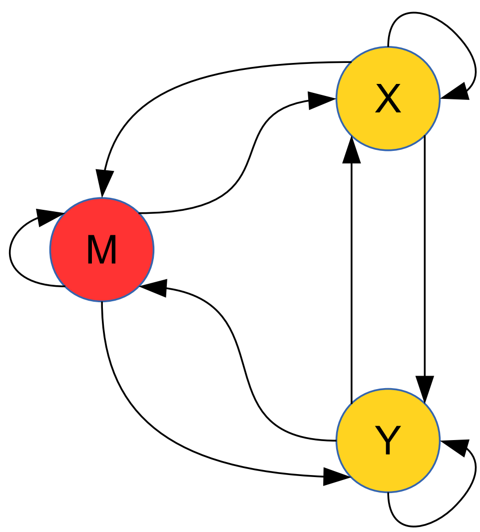
Ak máme oanotované trénovacie sekvencie, môžme z nich parametre odvodiť frekvenčnou analýzou. Emisie získame tak, že vyfiltrujeme symboly s príslušným stavom a spočítame frekvencie pre každý stav zvlášť a tranzície získame tak, že pre každý stav spočítame frekvencie nasledujúcich stavov. Tento postup sa volá *metóda maximálnej vierohodnosti (v angličtine maximum likelihood estimation)*. [DEKM98, Wik14]

1.6.4 Párové zarovnávanie pomocou HMM

ToDo: mic odporúča rovno popísať párové hmm a aj viterbiho rovno naňom, takže to asi prepíšem

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. 1.6). Tento automat vieme previesť na HMM.

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



Obr. 1.7: Párový HMM pre zarovnávanie sekvencií

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

Parametre modelu môžme 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]

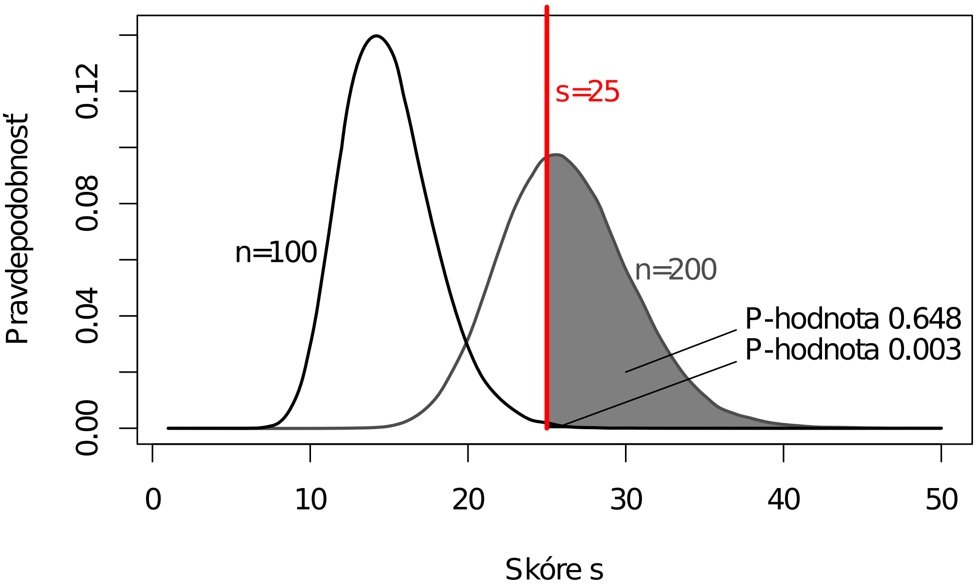
1.7 Štatistická významnosť zarovnania

Smith-Watermanov algoritmus nájde najlepšie lokálne zarovnanie, pre ľubovoľné dve sekvencie. Treba však rozhodnúť, či je zarovnanie dostatočne vierohodné na to, aby 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 takouto (a nižšou) 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 približne rovnaké, teda taký istý prah môžme použiť aj na E-hodnotu.



Obr. 1.8: 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 ploche 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 môžme rozlišovať dva typy modelov - *generatívny model* a *diskriminačný model*.

Konvenčné techniky odhadu pre zarovnávania 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 zľahčili 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 diskriminačné učenie (SVM, RandomForest) zvyčajne produkuje oveľa presnejšie pravidlá ako generatívne učenie(HMM, naive Bayes classifier). [YJEP07] Môže to byť vysvetlené tým, že $P(Z|X, Y)$, je už vhodné na vyhodnotenie optimálnej predikcie

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

[YJEP07]
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 trénovanie Support Vector Machine (SVM) dá pozerať ako na riešenie tohto problému. V článku sa zaoberajú použitím *Structural SVM* algoritmu na zarovnávanie proteínových sekvencií. Diskriminačné učenie umožňuje zahrnutie množstva dodatočnej informácie – státisíce parametrov. Navyše SVM umožňuje trénovanie 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 diskriminačného učenia, avšak narozdiel od [YJEP07] neumožňuje použitie účelových funkcií.

3 Klasifikácia na základe lokálnej informácie

3.1 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. 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.

InDel klasifikátor určuje s akou pravdepodobnosťou má byť pozícia v príslušnej sekvencii zarovnaná s medzerou. 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.

Tieto pravdepodobnosti sú akousi mierou istoty daného klasifikátora, a keďže tieto 2 klasifikátory sú nezávislé, súčet ich výstupov nemusí byť jedna.

3.2 Vstupné dáta

V práci sme vyskúšali a porovnali viacero typov vstupných dát a porovnali sme ako dobre sa klasifikátor na týchto dátach učí. Všetky typy dát sú založené na okne okolo daných pozícií, ktoré je definované v nasledujúcej sekcii.

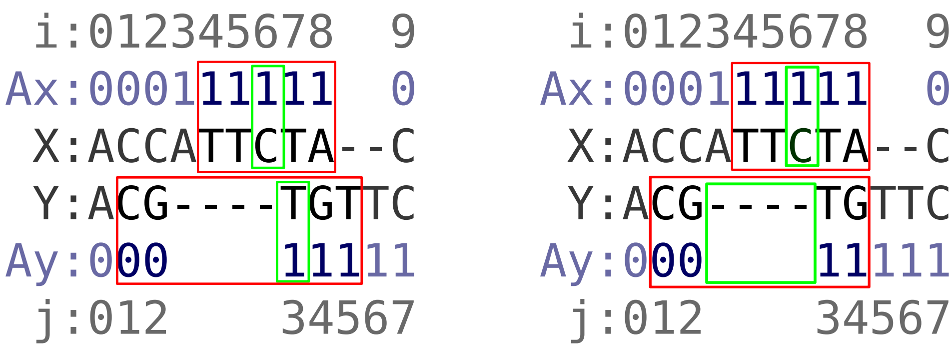
Na porovnanie typov dát sme používali dve miery – dôležitosť atribútov a úspešnosť klasifikátora.

3.2.1 Definícia okna

Ako vstupné dáta dostane klasifikátor okolie okolo daných pozícií. Toto okolie budeme volať *okno*. Okno veľkosti w pozostáva z $2w$ blokov veľkosti $k = (1 + \#\text{anotácií})$.

Majme teda dve sekvencie, $X = x_1x_2 \ldots x_n$ a $Y = y_1y_2 \ldots y_n$ a pozície i a j. Pri Match klasifikátore okno veľkosti w obsahuje $x_{i-w/2} \ldots x_i \ldots x_{i+(1+w)/2}$, $y_{j-w/2} \ldots y_j \ldots y_{j+(1+w)/2}$ a všetky anotácie príslušných báz. (Obr. 3.1a)

Pri InDel klasifikátore používame tiež 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. Predpokladajme teraz, že X je inzert sekvencia. Okno Indel klasifikátora veľkosti w obsahuje $x_{i-w/2} \ldots x_i \ldots x_{i+(1+w)/2}$, $y_{j-w/2} \ldots y_j \ldots y_{j+(1+w)/2-1}$ a všetky anotácie príslušných báz. (Obr. 3.1b)



(a) Match klasifikátor (b) InDel klasifikátor

Obr. 3.1: Okno klasifikátora pre pozície $i = 6$ a $j = 3$

3.2.2 Typ dát č. 1 - okno bez úpravy

Ako prvý typ dát sme zobrali okno tak ako sme ho definovali v predošlej sekcii (3.2.1). Dáta obsahujú priamo všetky bázy a anotácie tak ako sú v okne.

Na obrázku 3.2 si môžme všimnúť, že klasifikátory sa zamerali najmä na bázy a anotácie skoro nebral do úvahy. Toto správanie zodpovedá tomu, že v praxi bázy majú podstatne väčší význam pri zarovnávaní sekvencií.

Na obrázku 3.3 je graf distribúcie výstupu klasifikátora pre pozitívne a negatívne príklady. Môžme si všimnúť, že klasifikátory sa natrénovali dobre – teda pozitívne príklady (modrá) sa nachádzajú v ľavej časti a negatívne (zelená) sa nachádzajú v pravej časti. Pri Match klasifikátore (obr. 3.3a) dáva klasifikátor dokonca pre väčšinu pozitívnych príkladov hodnoty blízke 1 a naopak pre negatívne dáva najviac hodnoty blízke 0. InDel klasifikátor (obr. 3.3b) má výsledky trochu horšie, čo je celkom pochopiteľné, pretože pri medzerách sa pozitívne príklady identifikujú ťažšie.

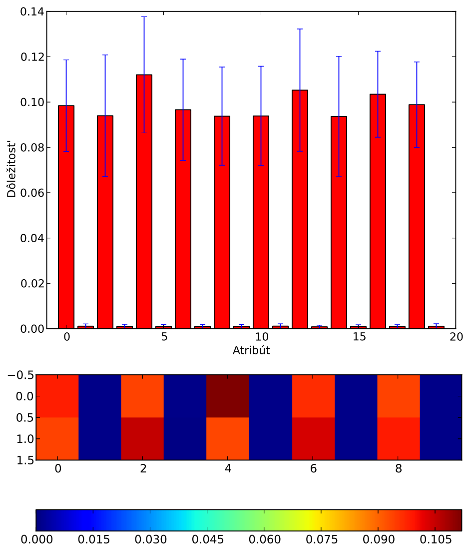
Celková úspešnosť Match klasifikátora bola na trénovacej vzorke 93,07% a na testovacej vzorke 83,57%. Úspešnosť Indel klasifikátora bola na trénovacej vzorke 88,51% a na testovacej 74,13%.

3.2.3 Typ dát č. 2 - zhody v stĺpcoch okna

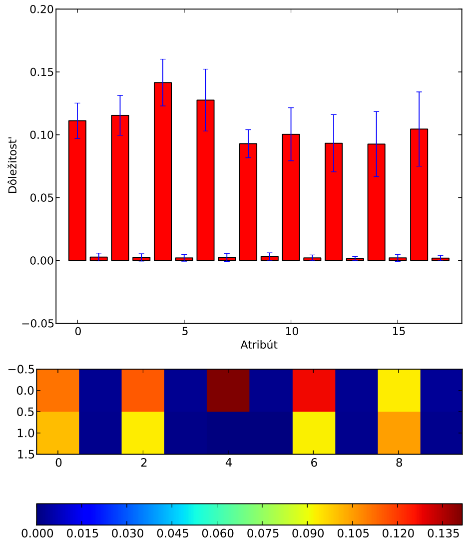
Druhý typ dát obsahuje aktuálnu bázu spolu s jej anotáciami a navyše pole veľkosti $k * w$, ktoré má na i-tom mieste 1 ak $okno_X[i] = okno_Y[i]$, ináč 0. Pričom w je veľkosť okna, k je veľkosť bloku, $okno_X$ je časť okna zodpovedajúca X sekvencii a $okno_Y$ zodpovedá Y-ovej časti okna. V Indel klasifikátore je jedna malá zmena - pozícia 3 aj 4 v x-ovej sekvencii sú porovnávané s pozíciou 3 v y-ovej a pozícia 5 v x-ovej sa porovnáva s pozíciou 4 v y-ovej. Pozíciu 3 v y-ovej sekvencii sme zopakovali pre to, že sme experimentom zistili, že pre klasifikátor je dôležitá – vidno to aj na obrázku 3.4b.

Pri tomto type dát to podľa obrázka 3.4 dopadlo veľmi podobne ako pri type popísanom v predchádzajúcej sekcii (3.2.2). Pri Indel klasifikátore (obr. 3.4b) sa nám potvrdila naša hypotéza, že vzťah pozície 3 v x-ovej a y-ovej sekvencii je dôležitý. Zhoda na týchto pozíciách totiž pomáha identifikovať negatívne príklady. Zaujímavosťou v tomto prípade je, že atribúty z pravej časti okna klasifikátoru neprišli zaujímavé.

Rovnako ako v prípade dát typu 1, aj pri tomto type dát vedeli klasifikátory rozlíšiť



(a) Match klasifikátor



(b) InDel klasifikátor

Obr. 3.2: **Dôležitosť atribútov pre typ dát č. 1** - hodnoty sú normalizované aby súčet bol 1, modrý pásik označuje štandardnú odchýlku cez jednotlivé stromy v Random foreste. Pod grafom je tepelná mapa pre lepšiu vizualizáciu. Okno je veľkosti 5, takže aktuálne sa pýtame na 3tie pozície v okne t.j. bázy x_3, y_3 a ich anotácie ax_3 ay_3 (resp. v InDel klasifikátore len bázu x_3 a anotáciu ax_3) Atribúty v Match klasifikátore sú číslované nasledovne:

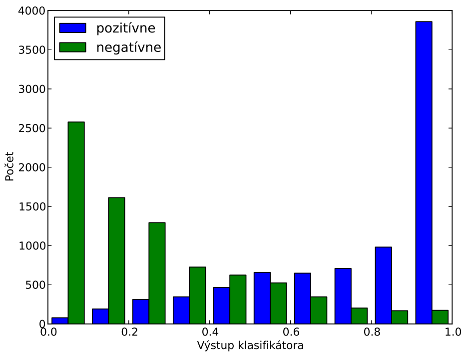
	0	1	2	3	4	5	6	7	8	9
0	x_1	ax_1	x_2	ax_2	x_3	ax_3	x_4	ax_4	x_5	ax_5
1	y_1	ay_1	y_2	ay_2	y_3	ay_3	y_4	ay_4	y_5	ay_5

a atribúty v InDel klasifikátore sú číslované takto:

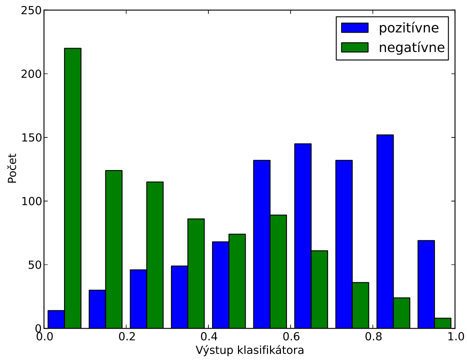
	0	1	2	3	4	5	6	7	8	9
0	x_1	ax_1	x_2	ax_2	x_3	ax_3	x_4	ax_4	x_5	ax_5
1	y_1	ay_1	y_2	ay_2	y_3	ay_3	y_4	ay_4		

,

pričom v tepelnej mape sú bázy a anotácie 3 a 4 posunuté doprava a medzi ne je vsunutá medzera.

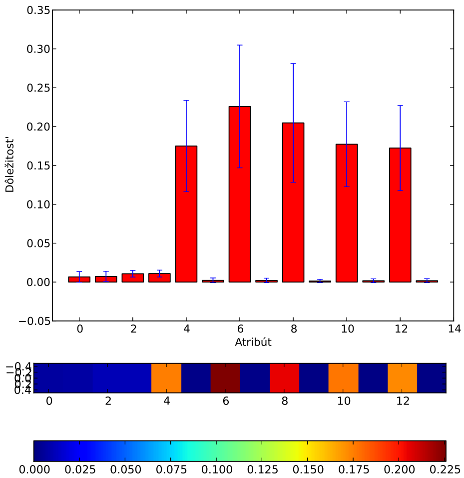


(a) Match klasifikátor

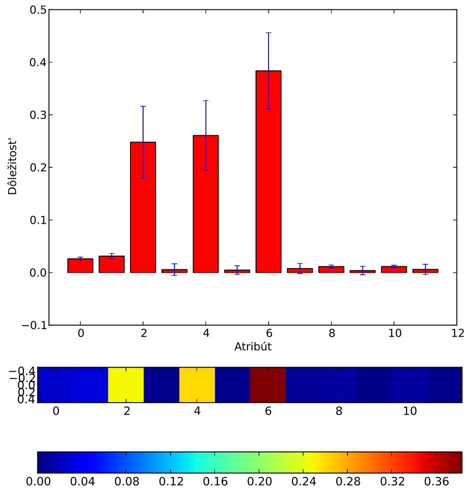


(b) InDel klasifikátor

Obr. 3.3: Distribúcia výstupu z klasifikátora pri type dát č. 1 – modré sú pozitívne príklady a zelené sú negatívne. Na x-ovej osi je výstup klasifikátora a na y-je počet inštancií, pre ktoré výstup z klasifikátora padol do daného chlievika

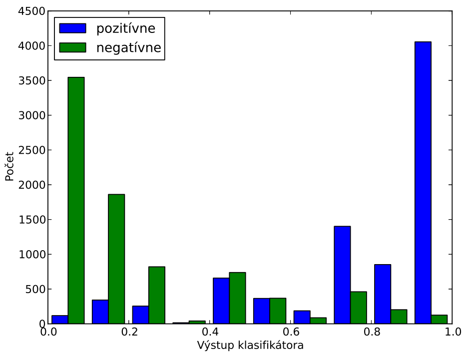


(a) Match klasifikátor

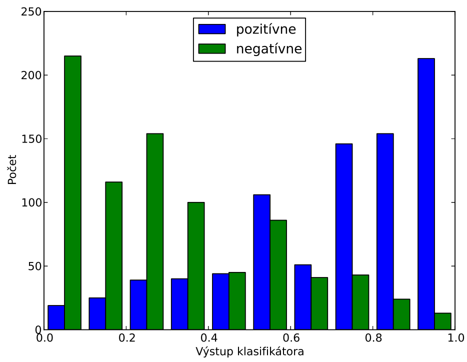


(b) InDel klasifikátor

Obr. 3.4: **Dôležitosť atribútov pre typ dát č. 2** - hodnoty sú normalizované aby súčet bol 1, modrý pásik označuje štandardnú odchýlku cez jednotlivé stromy v Random foreste. Pod grafom je tepelná mapa pre lepšiu vizualizáciu. Okno je veľkosti 5. Bázy a anotácie na ktoré sa pýtame sú na začiatku - pozície 0-3 (resp. v InDel klasifikátore pozície 0-1)



(a) Match klasifikátor



(b) InDel klasifikátor

Obr. 3.5: Distribúcia výstupu z klasifikátora pri type dát č. 2 – modré sú pozitívne príklady a zelené su negatívne. Na x-ovej osi je výstup klasifikátora a na y-je počet inštancií, pre ktoré výstup z klasifikátora padol do daného chlievika

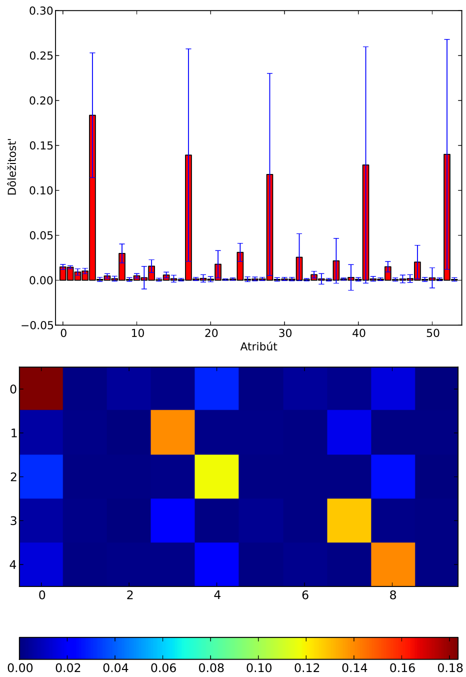
pozitívne a negatívne príklady. Pri Match klasifikátore (obr. 3.5a) však funkcia početnosti príkladov v závislosti od vzdialenosti od cieľovej hodnoty už nie je klesajúca ako to bolo v predošlom type dát (obr. 3.3a). Distribúcia Indel klasifikátora (obr. 3.5b) však vyzerá lepšie ako v predošlom prípade.

Celková úspešnosť Match klasifikátora na trénovacej množine bola 84,05% a na testovacej 84.31%. Úspešnosť Indel klasifikátora na trénovacej množine bola 77,17% a na testovacej 75,75%. Trénovacia chyba bola teda väčšia ale testovacia mierne menšia ako v predošlom prípade. Podobnosť trénovacej a testovacej chyby značí, že v tomto prípade nedošlo k pretrénovaniu, ale klasifikátor sa už nedokáže lepšie naučiť rozlišovať pozitívne a negatívne príklady.

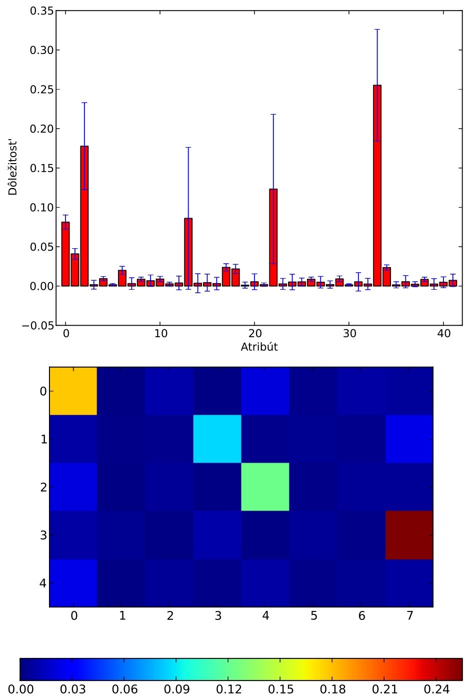
3.2.4 Typ dát č. 3 - matica zhôd v okne

Tretí typ dát je podobný ako typ č. 2 (sekcia 3.2.3), rozdiel je v tom, že teraz pole obsahuje nie len zhody po dvojiciach ale celú maticu zhôd. Teda opäť máme aktuálne bázy s anotáciami a pole má veľkosť $k * w^2$. Každý riadok sa skladá s jednotlivých blokov a v tabuľke v x-tom riadku, y-tom stĺpci a i-tom mieste v bloku je 1 práve vtedy keď $okno_X[x+i] = okno_Y[y+i]$.

Keďže v tomto prípade máme veľa atribútov na vyhodnotenie okrem obrázka 3.6, ktorý nie je dostatočne prehľadný, pomôžeme aj tabuľkou 3.1.



(a) Match klasifikátor



(b) InDel klasifikátor

Obr. 3.6: **Dôležitosť atribútov pre typ dát č. 3** - hodnoty sú normalizované aby súčet bol 1, modrý pásik označuje štandardnú odchýlku cez jednotlivé stromy v Random foreste. Pod grafom je tepelná mapa pre lepšiu vizualizáciu. Bázy na ktoré sa pýtame sa spolu s anotáciami nachádzajú na začiatku – pozície 0-3 (resp. 0-1 v Indel klasifikátore) a v tepelnej mape sme ich pre prehľadnosť vynechali. V Indel klasifikátore samozrejme chýba stredná pozícia v y-ovej sekvencii.

Poradie	atribút	dôležitosť	pozície v sekvencii (x, y)	báza/anotácia
1.	4	0.183560	(0, 0)	báza
2.	52	0.140056	(4, 4)	báza
3.	17	0.139246	(1, 1)	anotácia
5.	41	0.128264	(3, 3)	anotácia
4.	28	0.117699	(2, 2)	báza

(a) Match klasifikátor

Poradie	atribút	dôležitosť	pozície v sekvencii (x, y)	báza/anotácia
1.	33	0.255190	(3, 3)	anotácia
2.	2	0.177749	(0, 0)	báza
4.	22	0.123246	(2, 2)	báza
3.	13	0.086060	(1, 1)	anotácia
5.	0	0.081229	x:3	báza

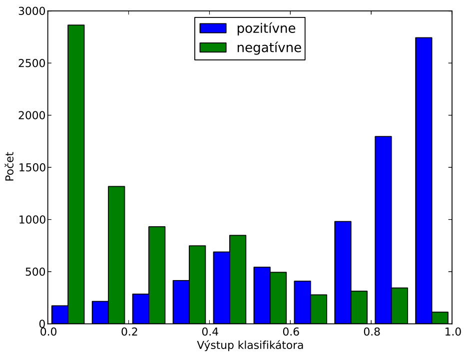
(b) Indel klasifikátor

Tabuľka 3.1: Najdôležitejšie atribúty pre typ dát č. 3

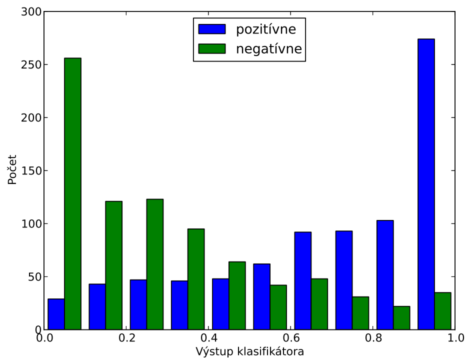
Všimnime si, že najdôležitejšie atribúty sú na uhlopriečke, čo zodpovedá typu dát č. 1 (sekcia 3.2.2), ibaže v tomto prípade sa nám vyskytli na niektorých pozíciách anotácie namiesto báz. Avšak ak sa nám tam vyskytla anotácia, bázu už klasifikátor nepovažoval za dôležitú a aj naopak. Indel klasifikátor navyše považoval za dôležitejšiu aj bázu na aktuálnej pozícii a jej anotáciu. Pri tomto type dát narozdiel od ostatných klasifikátor viac berie do úvahy aj anotácie.

Distribúcie výstupu klasifikátorov (obr. 3.7) majú očakávané vlastnosti, ale Match klasifikátor mal menej veľmi dobre klasifikovaných[1] a viac zle klasifikovaných dát ako s predchádzajúcimi typmi dát. To sa odrazilo aj na úspešnosti, kde Match klasifikátor dosiahol 80,44% na trénovacej množine a 79,79% na testovacej a Indel klasifikátor dosiahol 78,03% na trénovacej a 75,03% na testovacej množine.

[1]pozitívnych, ktoré majú hodnotu blízku 1 alebo negatívnych s hodnotou blízko 0



(a) Match klasifikátor



(b) InDel klasifikátor

Obr. 3.7: Distribúcia výstupu z klasifikátora pri type dát č. 3 – modré sú pozitívne príklady a zelené su negatívne. Na x-ovej osi je výstup klasifikátora a na y-je počet inštancií, pre ktoré výstup z klasifikátora padol do daného chlievika

3.2.5 Typ dát č. 4 - kombinácia 1 a 2

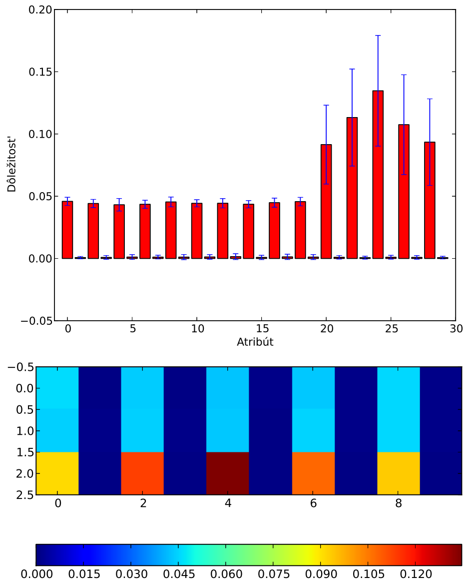
Posledný typ dát je kombináciou typov dát 1 a 2 (popísaných v sekciách 3.2.2 a 3.2.3). Dáta opäť obsahujú všetky bázy a anotácie a navyše sme pridali pole zhôd z dát typu 2. Táto informácia je síce redundantná a klasifikátor by si ju mal vedieť odvodiť aj sám, no predošlé experimenty ukázali, že by to mohlo pomôcť.

Na obrázku 3.8 si môžme všimnúť, že klasifikátory uprednostnili dáta typu 2 oproti dátam typu 1. Avšak aj bázy z dát typu 1 sa klasifikátoru zdajú dôležité. Pri InDel klasifikátore si môžme všimnúť, tento graf je vlastne zložením grafov z dát typu 1 (obr. 3.2b) a 2 (obr. 3.4b).

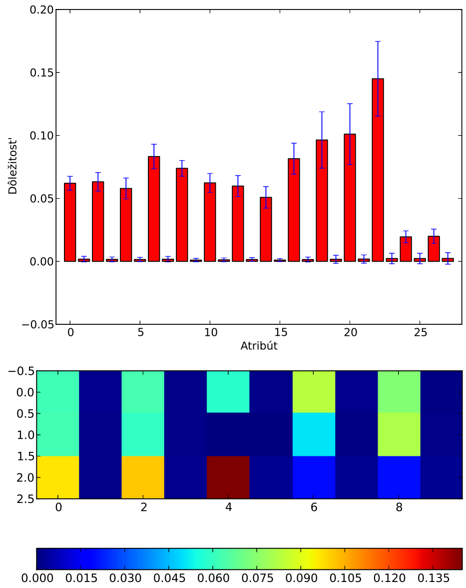
Podľa obrázku 3.9 je vidno, že spojenie 2 typov dát klasifikátoru pomohlo a dostali sme najlepšiu distribúciu spomedzi všetkých typov aj v Match aj v Indel klasifikátore. Aj úspešnosť je maximum z prezentovaných typov dát – match klasifikátor dosiahol 93,65% na trénovacej množine a 84,32% na testovacej. Indel klasifikátor dosiahol 88,78% na trénovacej a 46,46% na testovacej množine.

3.2.6 Zhrnutie

Ako sme už načrtli v predošlej sekcii (**??**), najlepšie dáta pre klasifikátor vyšli dáta typu 4. V tabuľke 3.2 je prehľad úspešností na testovacej množine pre Match aj Indel

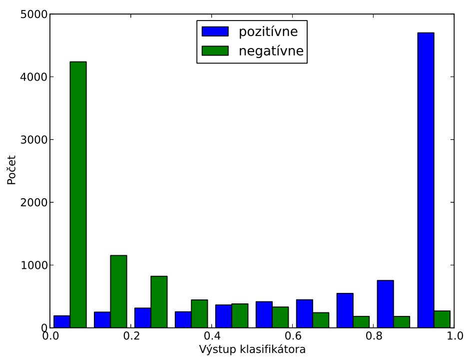


(a) Match klasifikátor

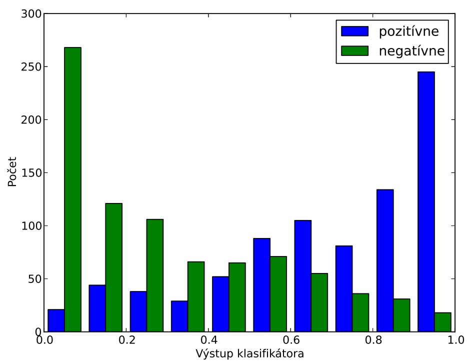


(b) InDel klasifikátor

Obr. 3.8: **Dôležitosť atribútov pre typ dát č. 4** - hodnoty sú normalizované aby súčet bol 1, modrý pásik označuje štandardnú odchýlku cez jednotlivé stromy v Random foreste. Pod grafom je tepelná mapa pre lepšiu vizualizáciu. Okno je veľkosti 5, takže aktuálne sa pýtame na 3tie pozície v okne t.j. bázy x_3, y_3 a ich anotácie ax_3 ay_3 (resp. v InDel klasifikátore len bázu x_3 a anotáciu ax_3)



(a) Match klasifikátor



(b) InDel klasifikátor

Obr. 3.9: Distribúcia výstupu z klasifikátora pri type dát č. 4 – modré sú pozitívne príklady a zelené su negatívne. Na x-ovej osi je výstup klasifikátora a na y-je počet inštancií, pre ktoré výstup z klasifikátora padol do daného chlievika klasifikátory.

Typ dát	Match	Indel
1	83,57%	74,13%
2	84,31%	75,75%
3	79,79%	75,03%
4	84,32%	76,46%

Tabuľka 3.2: Úspešnosť Match a Indel klasifikátorov pri rôznych spôsoboch predspracovania dát

Úspešnosť Match aj InDel klasifikátora je najlepšia a aj distribúcia má požadované vlastnosti. Preto sme sa rozhodli použiť tento typ dát pre naše experimenty.

3.3 Trénovanie

Pre oba typy klasifikátorov sme sa snažili nájsť vhodné pozitívne a negatívne trénovacie príklady, pričom z oboch sme do trénovacej množiny zahrnuli rovnako, aby bola trénovacia množina vyvážená.

3.3.1 Výber pozitívnych a negatívnych príkladov pre Match klasifikátor

Match klasifikátor sme chceli natrénovať tak, aby pre pozície, ktoré majú byť pri sebe, dával hodnoty blízke 1 a pre pozície ktoré k sebe byť zarovnané nemajú dával hodnoty blízke 0. Ako pozitívne príklady sme teda vybrali pozície z trénovacích sekvencií, ktoré boli zarovnané k sebe. Ako negatívne príklady sme najskôr zvolili náhodné pozície, tak aby sa nám nestalo, že by boli k sebe zarovnané. Toto sa však ukázalo ako nedostatočné riešenie, pretože sa mohlo ľahko stať, že sa k sebe dostali pozície s okolím, ktoré boli od seba pomerne ďaleko a tak sa mohlo ľahko stať, že v inej sekvencii, by mohli byť zarovnané k sebe. Preto sme sa rozhodli pristúpiť k inému spôsobu výberu negatívnych vzoriek.

Počas zarovnávania sa väčšinou lokálne rozhodujeme, či zarovnať dané pozície k sebe, alebo vložiť medzeru. Ak by sme vložili medzeru, nastal by posun v jednej zo sekvencií.

Rozhodli sme sa teda sa týmto inšpirovať a vyrábať negatívne príklady posunom zarovnaných pozícií. Pre každú zarovnanú pozíciu si najskôr rovnomerne náhodne vyberieme sekvenciu, ktorú budeme posúvať a potom z exponenciálneho rozdelenia náhodne vyberieme veľkosť posunu. Presný vzťah pre výpočet posunu Δ je

$$\Delta = (2D - 1) \cdot (1 + \lfloor S \rfloor) \qquad D \sim Alt(0.5),\, S \sim Exp(0.75),$$

kde prvý člen nám určuje smer posunu (čo je to isté ako: ktorá sekvencia sa bude posúvať) a druhý člen určuje veľkosť posunu - chceme celočíselnú hodnotu ≤ 1.

3.3.2 Výber pozitívnych a negatívnych príkladov pre Indel klasifikátor

Indel klasifikátor sme chceli natrénovať tak aby pre miesta, ktoré majú byť zarovnané s medzerou dával čo najvyššie hodnoty a pre miesta, ktoré nemajú byť zarovnané k medzere dával čo najnižšie hodnoty. Ako pozitívne príklady sme sa teda rozhodli vyberať pozície, ktoré boli v trénovacích sekvenciách zarovnané k medzere. A ako negatívne príklady sme vybrali pozície, ktoré boli zarovnané k sebe - teda tie isté, čo sme pri trénovaní Match klasifikátora (sekcia 3.3.1) považovali za pozitívne, akurát v tomto prípade mala jedna zo sekvencií okno skrátené o 1, teda akoby bola medzera pred bázou, s ktorou je aktuálne uvažovaná pozícia zarovnaná.

3.4 Random Forest

ToDo: chcem to tu, alebo niekde inde?

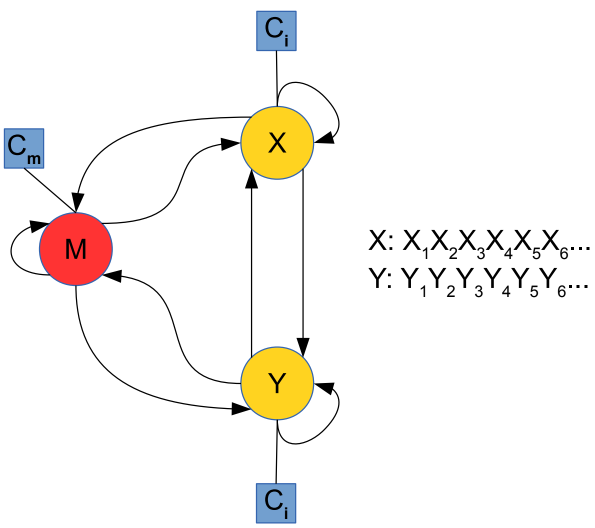
ToDo:

4 Modely

V sekcii 1.6 sme si zadefinovali HMM pre zarovnávanie sekvencií (obr. 1.7). V našom riešení sme predstavili 2 modifikácie pôvodného HMM na zakomponovanie dodatočnej informácie, pričom sme využili klasifikátory. V oboch modeloch sú klasifikátory rovnaké, aj s rovnakým postupom trénovania. Líši sa len trénovanie samotného HMM a architektúra modelu.

4.1 Model s klasifikátorom ako emisiou

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



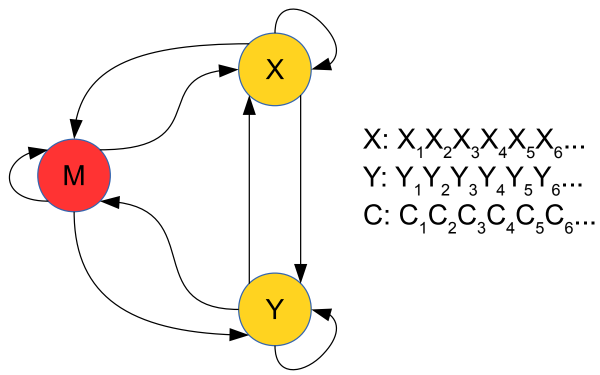
Obr. 4.1: Model s klasifikátorom ako emisiou

Problémom tohto modelu je, že výstup klasifikátora nezodpovedá emisným pravdepodobnostiam, 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 nevadí, avšak o takomto modeli už nemôžme 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 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 akúsi pomôcku pre náš zarovnávač.



Obr. 4.2: 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.

4.2.1 Trénovanie modelu

Pri tomto modeli stojí za zmienku aj spôsob trénovania. Trénovali aj tranzície aj emisie, pričom základný princíp ostáva taký, ako sme si ho popísali v sekcii 1.6.3.

Zaujímavé je trénovanie emisií klasifikátora.

V Match stave modelujeme 3 náhodné premenné – X, Y a C, pričom tieto náhodné premenné sú nezávislé. Teda platí $P(X \cap Y \cap C) = P(X \cap Y)P(C)$. $P(X \cap Y)$ poznáme z frekvenčnej tabuľky a potrebujeme ešte zistiť $P(C)$. Môžme použiť nasledujúcu vetu:

Veta 4.2.1 (Veta o úplnej pravdepodobnosti)**.** *Nech $A_1 \dots A_n$ tvoria rozklad univerza Ω a nech B je udalosť, potom*

$$P(B) = \sum_{i=1}^{n} P(B|A_i)P(A_i)$$

Máme teda

$$P\left[C = c\right] = \sum_{\forall x \in X, y \in Y} P\left[C = c | X = x \wedge Y = y\right] P\left[X = x \wedge Y = y\right],$$

pričom druhý člen poznáme a $P\left[C = c | X = x \wedge Y = y\right]$ už vieme ľahko dopočítať. Keď máme fixnuté dve bázy, vieme vybrať z trénovacej sekvencie všetky pozície, kde sú zarovnané tieto bázy a vypočítať distribúciu C.

Pre Indel stav postupujeme analogicky, ibaže namiesto $P(X \cap Y)$ máme buď $P(X)$ alebo $P(Y)$ podľa toho, ktorý stav práve počítame.

ToDo: preco som toto cele vlastne robil

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 <0, 1>.

5 Experimenty

5.1 Doplnkové informácie k sekvenciám a zdroje dát

5.2 Metódy vyhodnocovania výsledkov

ToDo: klasifikator

ToDo: zhoda

ToDo: traniztivita

5.3 Klasifikátor

5.4 Modely

ToDo: veeela tabuliek

5.5 Experimenty s biologickými dátami

ToDo: asi by sa patrilo to tu mať

6 Implementácia

ToDo: daky pokec o tom ze som to pisal v pythone a preco je python super

6.1 Použité knižnice

6.1.1 Realigner

ToDo: cosi k micovmu kodu - mozno referencia na jeho dizertacku + mozno nejaky pokec k tomu ze co sme pouzili a mozno ako to funguje

6.1.2 Pythonové knižnice

ToDo: python kniznice - najma scikit-learn, numpy, track

6.2 Triedy na zarovnávanie s klasifikátorom

6.2.1 Predspracovanie dát

ToDo: DataPreparers, class diagram

6.2.2 Zovšeobecnenie klasifikátora

ToDo: PairClassifier

6.2.3 HMM stavy s klasifikátorom

ToDo: Classifier state...

6.3 Pomocné programy

6.3.1 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.

Algoritmus

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

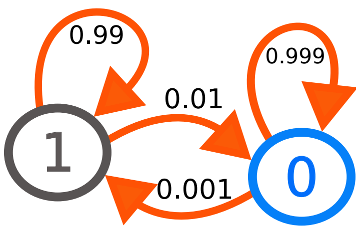
Okrem toho simulátor vygeneruje dve sekvencie, pričom prvú vyrobí náhodne a druhú odvodí z nej pomocou mutácie a inzercie/delécie. V našom prípade sme inzerciu vynechali a simulujeme ju ako deléciu v druhej sekvencii. Pri odvodzovaní bude používať aj informáciu o tom, ktorá časť je gén a ktorá nie.

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.

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 môžme opäť začať gén atď.



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

Generovanie robíme pomocou *Markovovskej reťaze (Markov Chain)* obr. 6.1. Generujeme podľa aktuálneho stavu a v každom kroku sa podľa pravdepodobnosti rozhodneme či sa prepneme 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.

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

Simulácia delécie

Deléciu simulujeme opäť pomocou Markovovskej 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 `'-'`.

Využitie

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

ToDo: vieme ho prisposobovat a merat na nom korektnost a uzitocnost modelu, alebo nieco na ten styl

6.3.2 Trénovanie modelov

ToDo: modeltraining.py

6.3.3 Testovanie klasifikátora

ToDo: random_forest_evaluation.py

6.4 Použitie

ToDo: v kratkosti o tom ako to vobec spustit so svojimi sekvenciami a modelmi...

ToDo: mozno sem dat aj moznosti rozsirenia

ToDo: konfiguracia - constants.py a config.py

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.

[NJ02] Andrew Y Ng and Michael I Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in neural information processing systems*, 2:841–848, 2002.

[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 subsequences. *Journal of Molecular Biology*, 147(1):195 – 197, 1981.

[Wik14] Wikipedia. Maximum likelihood. `http://http://en.wikipedia.org/wiki/Maximum_likelihood`, 2014. [Online; accessed 17-Apr-2014].

[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.