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.

Bratislava, 2014

Bc. Michal Hozza

Poďakovanie...

Bc. Michal Hozza

Abstrakt

Slovenský abstrakt

Kľúčové slová: ...

Abstract

English abstract

Key words: ...

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Najnovšie technológie sekvenovania DNA produkujú stále väčšie monožstvo 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 pravdepodobostnom modeli, pričom sa snažia nájsť zarovnanie s čo najväčšou pravdepodobnosťou. Algoritmy sú zvyčajne založené na dynamickom programovaní. Sú buď deterministické, ktoré s určitosťou nájdu najpravdepodbnejšie zarovnanie, ale za cenu kvadradického 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*. Náš 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 vacsi pocet 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, využitu natrénovaného klasifikátora na zarovnávanie nových sekvencií aj pri *prezarovnaní* už existujúcich zarovnaní a implementujeme univerzálny framework, ktorý umožní využívať náš klasifikátor pri rôznych zarovnaniach.

Kapitola 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 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.

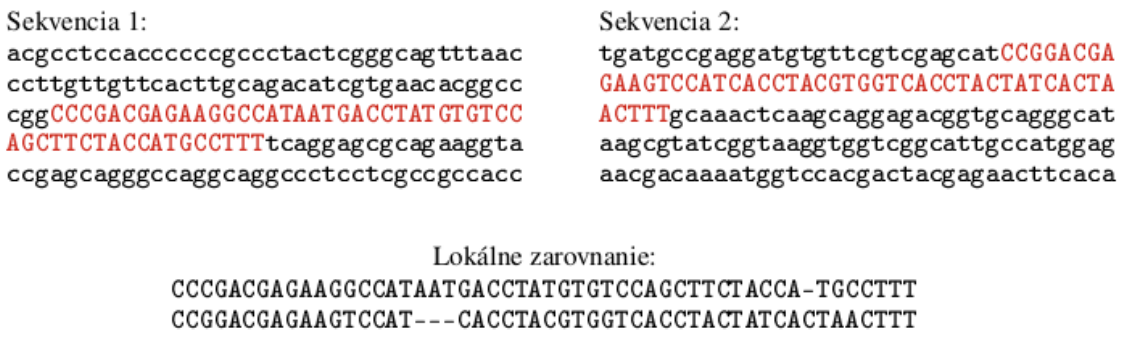
Počas evolúcie dvoch homologických sekvencií nastane veľa *inzercií*, *delécií* a *substitúcií*, preto predtým ako môžeme začať porovnávať sekvencie, ich musíme zarovnať tak, aby homologické časti sekvencií boli na rovnakom mieste v zarovnaní. [DEKM98, BV11]

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:

1. Aké typy zarovnávania by sme mali uvažovať



Obr. 1.1: Lokálne zarovnanie

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 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 poreťazcov $x_i \dots x_j$ 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 štrukturu 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 záviseť od toho aké bázy sú v danom stĺpci zarovnania. Na to môžme 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]

1.4.2 Afínne skórovanie medzier

V jednoduchom skórovaní sme dávali za pomčlku 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*. [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

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

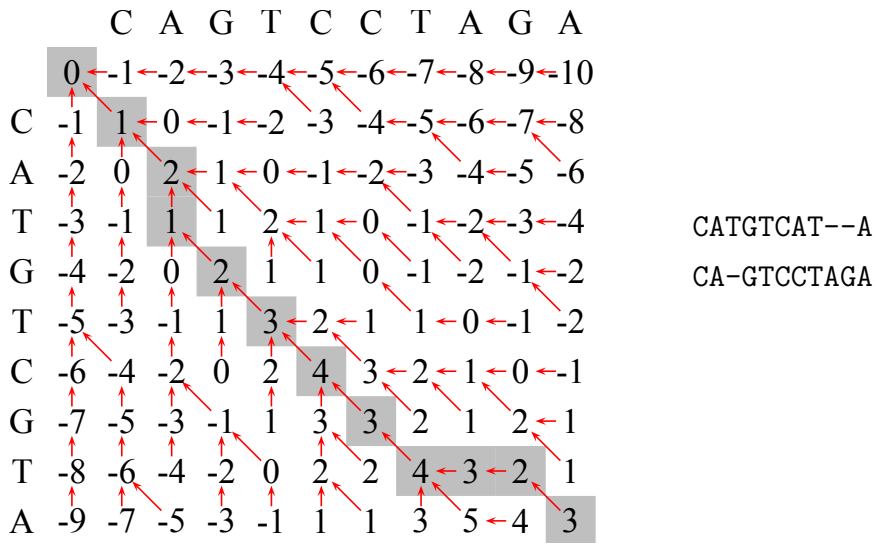
možných globálnych zarovnaní. Čiže nie je možné v rozumnom čase nimi prejsť.

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 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. Budeme používať jednoduché skórovanie: $+1$ za zhodu, -1 za nezhodu alebo pomlčku.

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.



Obr. 1.2: Globálne zarovnanie

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 $x_1x_2 \ldots x_i$ a $y_1y_2 \ldots y_j$. 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 buď $+1$ ak $x_i = y_j$ alebo -1 ináč. Ak by sme posledný stĺpec zmazali, dostali by sme zarovnanie $x_1x_2 \ldots x_{i-1}$ a $y_1y_2 \ldots y_{j-1}$, 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 $x_1x_2 \ldots x_{i-1}$ a $y_1y_2 \ldots y_j$, 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 vypĺňať po riadkoch, pričom každé políčok vieme vypočítať z troch políčok, ktoré už sú vypočítané.

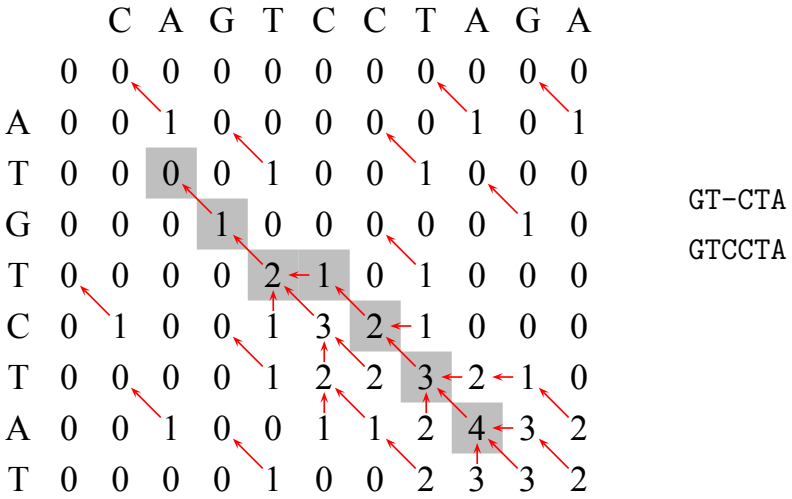
Ak nás zaujíma aj zarovnanie – nie len jeho skóre – vieme si pre každé políčko zapamätať ktorá z 3 možností dosiahla maximálnu hodnotu (červené šípky na Obr. 1.3).

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 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. 1.3: Lokálne zarovnanie

Algoritmus pre lokálne zarovnania sa líši len v niekoľkých malých detajloch. Opäť vypĺňame maticu A, s tým, že v $A[i, j]$ bude najvyššie skóre lokálneho zarovnania medzi sekvenciami $x_1x_2 \ldots x_i$ a $y_1y_2 \ldots y_j$, 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:

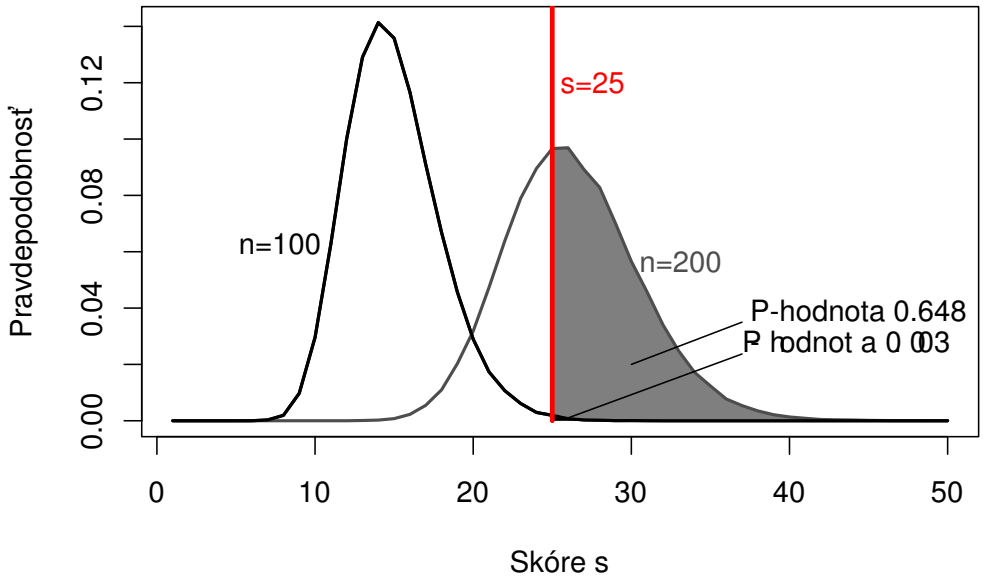
$$A[i,j] = \max \begin{cases} 0 \\ A[i-1, j-1] + s(x_i, y_j) \\ A[i-1, j] - 1 \\ A[i, j-1] - 1 \end{cases}$$

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

1.6 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)*.



Obr. 1.4: 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.

P-hodnota zarovnania je pravdepodobnosť, že medzi nádhodne 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 prebližne rovnaké, teda taký istý prah môžme použiť aj na E-hodnotu.

Kapitola 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 $X = x_1x_2 \ldots x_n$, $Y = y_1y_2 \ldots y_m$ 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 ovela 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 trénovanie Support Vector Machine (SVM) dá pozerať 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 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 diskriminatívneho učenia, avšak narozdiel od [YJEP07] neumožňuje použitie účelových funkcií závislých na aplikácii.

Kapitola 3

Doplnkové informácie k sekvenciám

Kapitola 4

Random forest[Draft]

V tejto kapitole si popíšeme ako funguje klasifikátor *Random forest*, v čom spočívajú jeho výhody a prečo sme si ho vybrali.

4.1 Klasifikačné stromy

Keďže Random forest je vlastne les *klasifikačných stromov (niekedy označované ako rozhodovacie stromy)*, je nevyhnutné, aby sme si najskôr povedali niečo o nich.

Klasifikačný strom je klasifikátor vybudovaný na základe trénovacej množiny, ktorý na základe vstupných údajov (vstupného vektora) predpovedá hodnotu výstupnej premennej. Klasifikátor sa skladá z *uzlov* a *listov*. V uzle sa nachádza rozdeľovacie kritérium, podľa ktorého sa vieme rozhodnúť do ktorej vetvy máme pokračovať. V listoch sa nachádzajú triedy, do ktorých sa vstupný vektor klasifikuje.

Klasifikácia vstupného vektora vyzerá nasledovne: prechádzame strom zhora na dol a postupne sa zaraďujeme do vetiev podľa rodzelovacích kritérií. Podľa toho, v ktorom liste skončíme, sa určí výstupná hodnota.

4.1.1 Trénovanie

TODO - strucne popisat rozne metody + odkazat na literaturu

4.2 Random forest

4.2.1 Klasifikácia

Random forest sa skladá z mnohých klasifikačných stromov. Pri klasifikácii nejakého vstupnéeho vektora sa predloží vstupný vektor každému stromu. Každý strom následne

vráti klasifikáciu a hovoríme, že stromy *hlasujú*. Random forest následne zvolí klasifikáciu podľa väčšiny z hlasov všetkých stromov v lese.

4.2.2 Trénovanie

Každý strom sa buduje nasledovne. Máme N trénovacích vektorov, pričom každý vektor sa skladá z M vstupných premenných (teda má rozmer M). Z nich vyberieme *náhodne s opakovaním* N vektorov, ktoré budú použité ako tréenovacia množina pre daný rozhodovací strom. Okrem toho vyberieme náhodne $m << M$ vstupných parametrov a len tie sa použijú pri trénovaní stromu. Každý strom je vybudovaný najviac ako sa dá - nie je žiadne orezávanie.

Chyba RF závisí od dvoch vecí

- korelácia
- sila

4.3 Dôležité vlastnosti Random forest-u

- Je najpresnejší spomedzi všetkých terajších klasifikačných algoritmov.
- Je efektívny aj na veľkých dátach.
- Dokáže obsiahnuť tisícky vstupných premenných bez ich vymazávania.
- Dáva odhad, ktoré premenné sú dôležité na klasifikáciu.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses.
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.
- It has methods for balancing error in class population unbalanced data sets.
- Generated forests can be saved for future use on other data.
- Prototypes are computed that give information about the relation between the variables and the classification.
- Počíta blízkosť medzi jednotlivími pármi prípadov, čo môže byť použité pri clusteringu a detekcii outlierov alebo ponúknuť zaujímavé pohľady na dáta.

- Predchádzajúce môže byť rozšírené aj na neolabelované dáta (bez informácie o triede), čo sa dá použiť na klasifikáciu, pohľady na dáta a detekciu outlierov bez učiteľa.
- Ponúka experimentálnu metódu na detekciu interakcie medzi premennými.

Kapitola 5

Návrh riešenia

Kapitola 6

Implementácia

6.1 Simulátor

Simulátor sĺuži 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 prisĺuchaju 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.

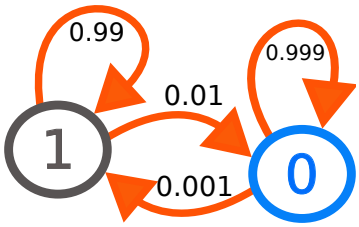
Okrem toho simulátor vygeneruje dve sekvencie, pričom prvú vyrobí náhodne a druhú odvodí z nej pomocou mutácie a inzercie/delécie. V našom prípade sme inzerciu vynechali a simulujeme ju ako deléciu v druhej sekvencii. Pri odvodzovaní bude používať aj informáciu o tom, ktorá časť je gén a ktorá nie.

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

Simulátor má vopred daných niekolko 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 môžme 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 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.

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.

Kapitola 7

Zhodnotenie úspešnosti

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.

[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.

[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.