

1.1. PODOBNOSŤ SEKVENCII, SEKVENCIONÁLNA HOZADNÁVA A ZERSEKVENCIEI

Sekvencia 1:

gagacccgcctaggtgaatatttagcagc
gattaaataccacgta**TATAAGGTGGACC**
GTTCTCGAGAGGTTCTTCCGGCAATGAC
GGCCAGAGCAAAGCCACGTgtaggactg
catacgcctctacgcctccactgacgcga
tgatgtggcgtggatctgtttgctcttg
tataggtcacggagacggctggtactgat
cccttcgggagtaaaaatataatgacat
ggcccaggcttcaggaggagttgtgcgg

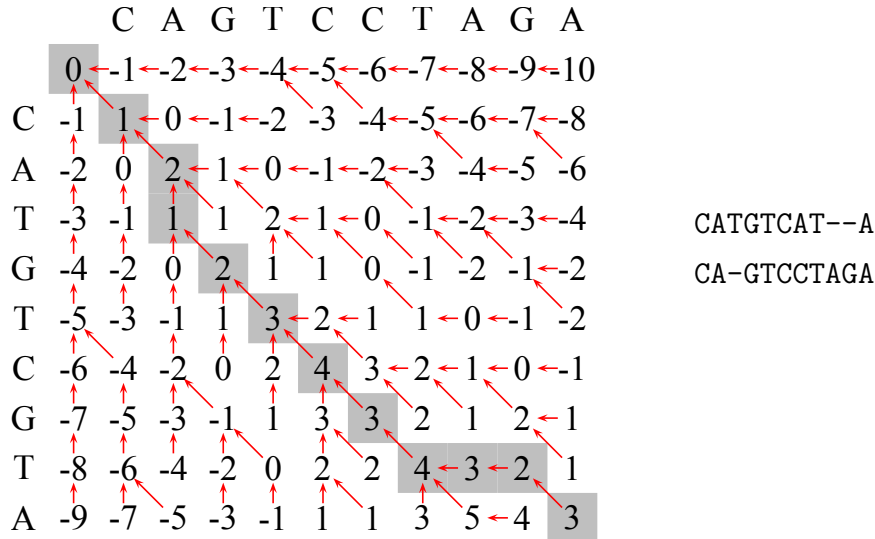
Sekvencia 2:

tgtacagcactgcaacgagcatctggggg
ttggttattccgatggcgctggacagcta
gcgacagtagttctcaggccttagtaga
aaggtgggaaccccc**TATGAGGTGACCG**
TTTCAGCGTGACTATAGACGTCATTGAAG
CAATATACAGGAACACCACCTacttagga
agggagttcgggtgcagtaaagcattctta
cctcagggcacggtagagaacactacaac
cagaatagcaacgtgatgcggcgactctc

Lokálne zarovnanie:

TATAAGGTGGACCGTT-----CCTCGAGAGGTTCTTCCGGCAATGGCCACGAGAGCAAAGCCACGT
TATGAGGTGACCGTTTTCAGCGTGACTATAGACGTCATTGAAGCAATATACAGG-----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



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.

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 $A[i, j]$:

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

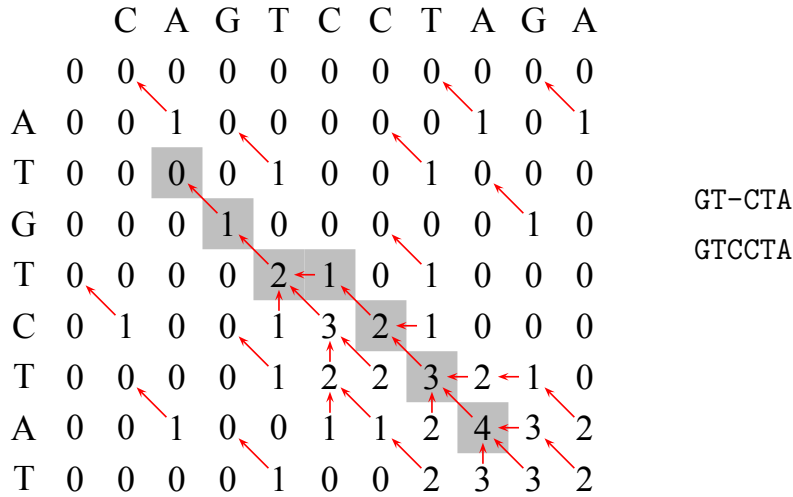
Maticu vieme vyplňať po riadkoch, pričom každé políčko vieme vypočítať z troch políčok, ktoré už sú vypočítané. Políčko $A[0, 0] = 0$ a krajné políčka potom vieme vypočítať ako $A[i, 0] = A[i-1, 0] - d$ a $A[0, j] = A[0, j-1] - d$

Ak nás zaujíma aj zarovnanie – nie len jeho skóre – vieme si pre každé políčko zapamätať ktorá z troch 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



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.

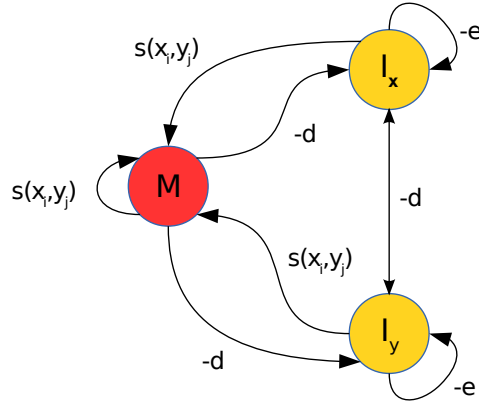
Algoritmus pre lokálne zarovnania sa líši len v niekoľkých malých detailoch. Opäť vyplníme maticu A , s tým, že v $A[i, j]$ bude najvyššie skóre lokálneho zarovnania medzi sekvenciami $x_1x_2 \dots x_i$ a $y_1y_2 \dots y_j$, ktoré buď obsahuje bázy x_i aj y_j , alebo je prázdne. Teda na ľubovoľnom mieste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

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

V tomto prípade sú všetky krajné políčka nulové.

Zarovnanie potom nájdeme tak, že začneme z políčka s maximálnym skóre, a postupne budeme prechádzať na políčka, z ktorých sme získali maximálnu hodnotu, kým nenarazíme na nulu.

Tieto vzťahy vieme popísať stavovým diagramom na obrázku 1.6:



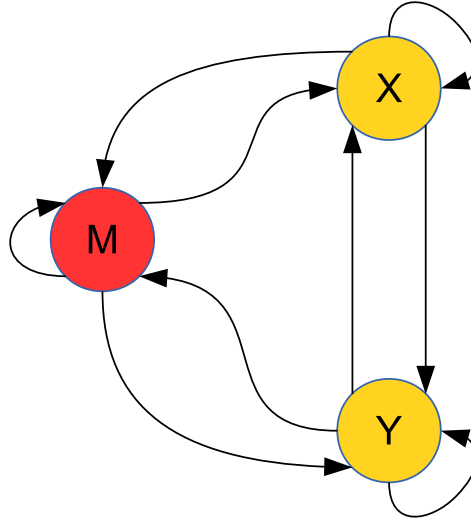
Obr. 1.6: Stavový diagram pre zarovnanie sekvencií - obsahuje *Match* (M), *InsertX* (I_x) a *InsertY* (I_y) stav a prechody medzi nimi spolu s ich cenou. Napríklad prechod z M do I_x znamená vloženie medzery do Y -ovej sekvencie a to penalizujeme $-d$

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 Markovovských modelov

Skrytý markovovský model (Hidden Markov Model, HMM) je generatívny pravdepodobnostný model, ktorý generuje náhodnú sekvenciu spolu s jej anotáciou (stavmi). HMM sa podobá na konečný automat. Skladá sa z konečného množstva stavov, prechodov medzi nimi a emisií. V každom kroku HMM vygeneruje a presunie sa do nejakého (aj toho istého) stavu. Generovanie symbolov aj presun medzi stavmi prebieha pravdepodobnostne. Konkrétny HMM je definovaný množinou stavov a nasledujúcimi distribúciami.

- distribúcia začiatočných stavov (pravdepodobnosť π_i , že HMM začne v stave i)
- distribúcia prechodov (pravdepodobnosť $a_{i,j}$, že HMM prejde zo stavu i do stavu j)
- distribúcia emisií (pravdepodobnosť $e_{i,x}$, že HMM v stave i vygeneruje symbol x)



Obr. 1.7: Párový HMM pre zarovnávanie sekvencií

Inicializácia (* označíme ľubovoľnú možnú hodnotu):

$$V[0, 0, M] = 1 \quad (1.1a)$$

$$V[i, 0, *] = 0 \quad \forall i \quad (1.1b)$$

$$V[0, j, *] = 0 \quad \forall j \quad (1.1c)$$

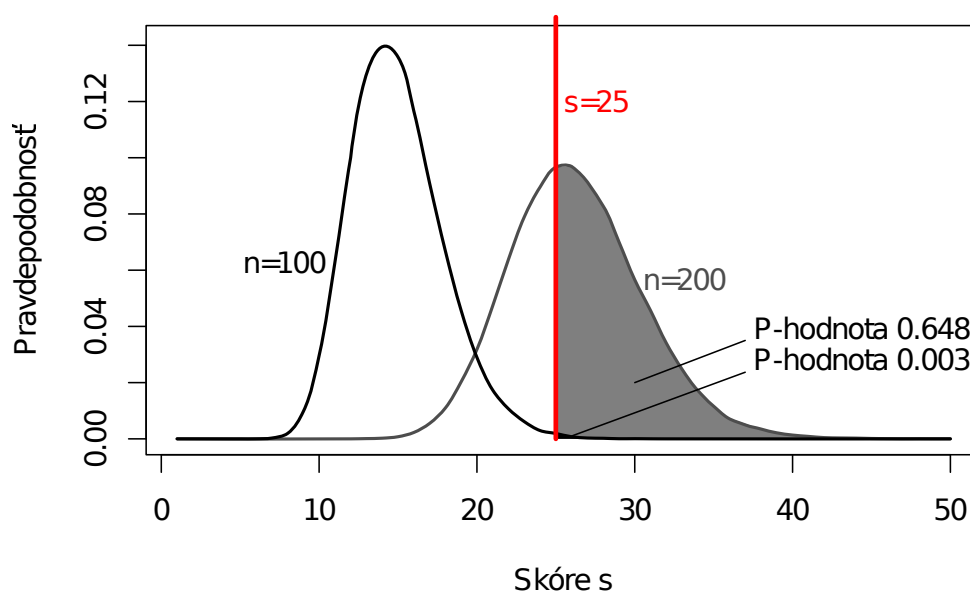
Rekurentné vzťahy:

$$V[i, j, M] = e_{M, (x_i, y_j)} \max \begin{cases} a_{M, M} V[i-1, j-1, M] \\ a_{I_x, M} V[i-1, j-1, I_x] \\ a_{I_y, M} V[i-1, j-1, I_y] \end{cases} \quad (1.2a)$$

$$V[i, j, I_x] = e_{I_x, x_i} \max \begin{cases} a_{M, I_x} V[i-1, j, M] \\ a_{I_x, I_x} V[i-1, j, I_x] \\ a_{I_y, I_x} V[i-1, j, I_y] \end{cases} \quad (1.2b)$$

$$V[i, j, I_y] = e_{I_y, y_j} \max \begin{cases} a_{M, I_y} V[i, j-1, M] \\ a_{I_y, I_y} V[i, j-1, I_y] \\ a_{I_x, I_y} V[i, j-1, I_x] \end{cases} \quad (1.2c)$$

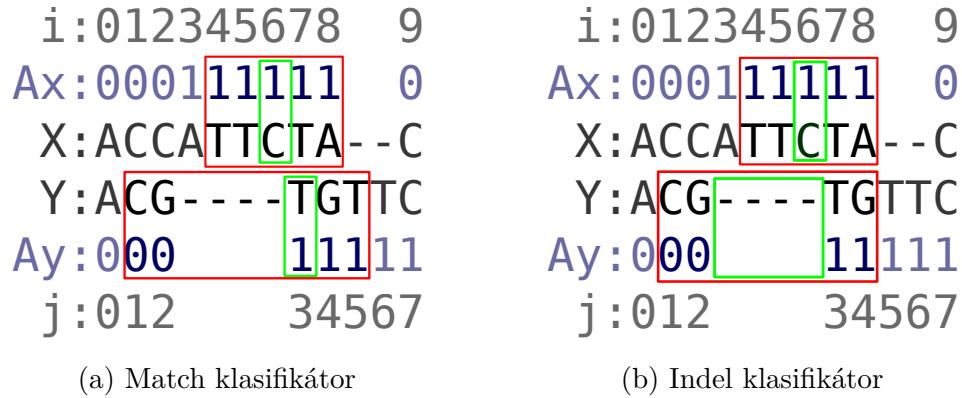
Algoritmus funguje takto: Nech n a m sú dĺžky sekvencií X a Y . Na začiatku na-
inicializuje hodnoty pre $V[0, *, *]$ a $V[*, 0, *]$ podľa 1.1 (* označíme ľubovoľnú možnú
hodnotu). Potom postupne pre všetky $i = 1 \dots n$, $j = 1 \dots m$ a $u \in \{M, I_x, I_y\}$ dyna-
micky vypočíta $V[i, j, u]$ podľa 1.2.



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.

Definícia okna

Okno veľkosti w pozostáva z $2w$ blokov veľkosti $k = (1 + \# \text{anotácií})$. Majme teda dve sekvencie, $X = x_1x_2 \dots x_n$ a $Y = y_1y_2 \dots y_n$ a pozície i a j . Pri Match klasifikátore 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. 2.2a). Pri Indel klasifikátore používame tiež dve pozície – prvá je pozícia bázy, na ktorú sa pýtame a druhá je pozícia medzery medzi dvoma bázami. Predpokladajme teraz, že X je sekvencia s bázou. Okno Indel klasifikátora 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. 2.2b).

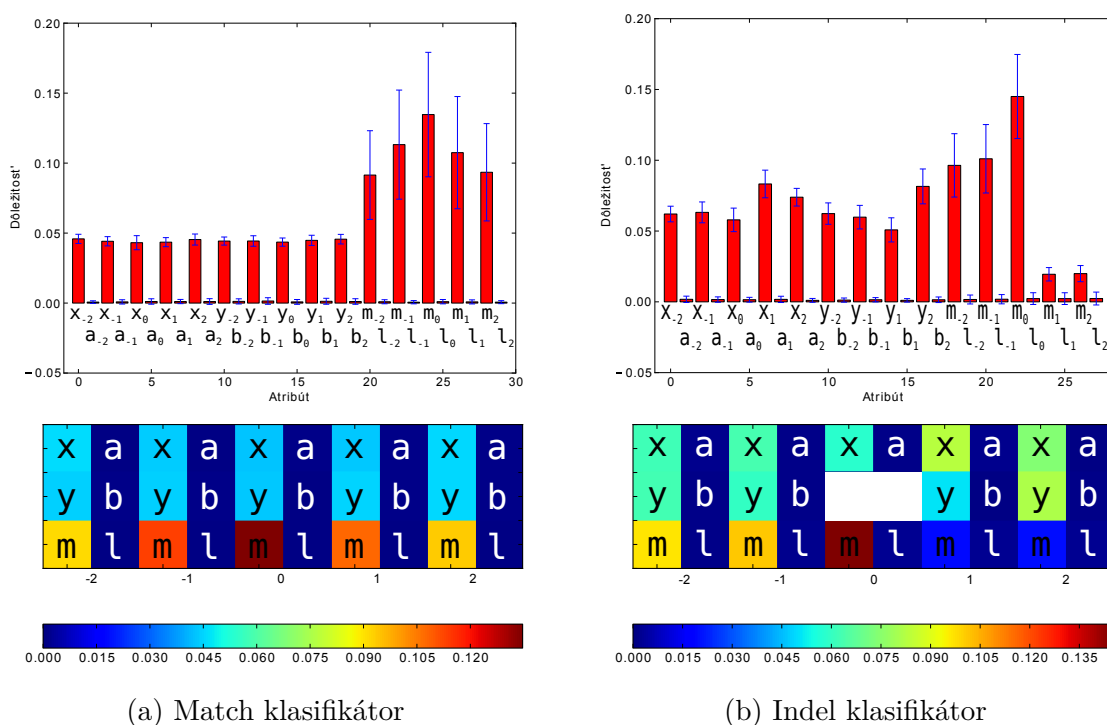
Obr. 2.2: Okno klasifikátora veľkosti 5 pre pozície $i = 6$ a $j = 3$ **Typ dát A - okno bez úpravy**

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

Typ dát B - 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 jedna ak $okno_X[i] = okno_Y[i]$, ináč nula (w je veľkosť okna, k je veľkosť bloku, $okno_X$ je časť okna zodpovedajúca sekvencii X a $okno_Y$ sekvencii Y). V Indel klasifikátore je jedna malá zmena: pozícia nula aj jedna v x -ovej sekvencii sú porovnávané s pozíciou jedna v y -ovej a pozícia dva v x -ovej sa porovnáva

Pri type dát C si môžeme všimnúť, že najdôležitejšie atribúty sú na uhlopriečke, čo zodpovedá typu dát B, 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 type dát C, na rozdiel od ostatných, klasifikátor viac berie do úvahy aj anotácie.

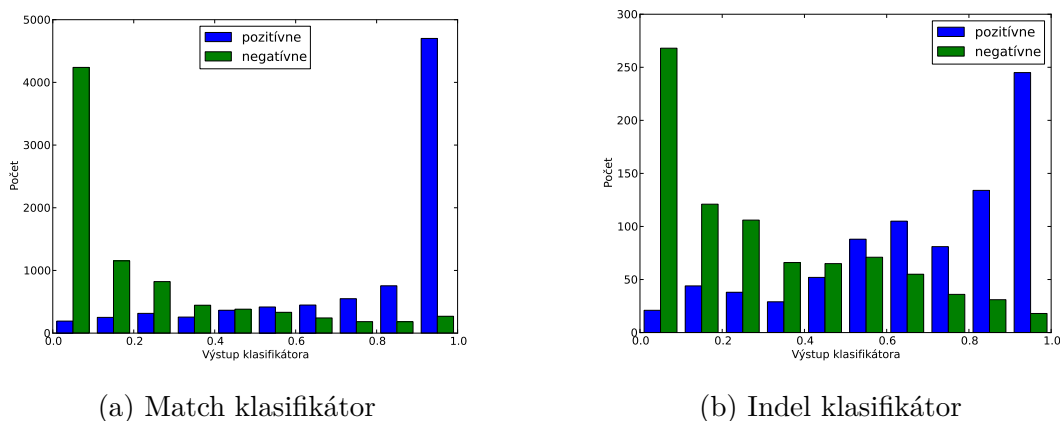


Obr. 2.3: **Dôležitosť atribútov pre typ dát D** - hodnoty sú normalizované, aby súčet bol jedna, modrý pásik označuje štandardnú odchýlku cez jednotlivé stromy v náhodnom lese. Pod grafom je tepelná mapa pre lepšiu vizualizáciu.

Pri dátach typu D sme dostali v prípade Match klasifikátora očakávanú distribúciu dôležitosti atribútov (obr. 2.3). Dáta typu B boli uprednostnené voči typu A, avšak aj bázy z typu A boli dôležité. Pri Indel klasifikátore sme dostali graf, ktorý je zložením grafov z dát typu A a B. Zaujímavosťou je, že pri tomto type dát má výraznejšiu úlohu práve zhoda na stredovej pozícii, teda pozície už nie sú také rovnocenné ako to bolo v type B.

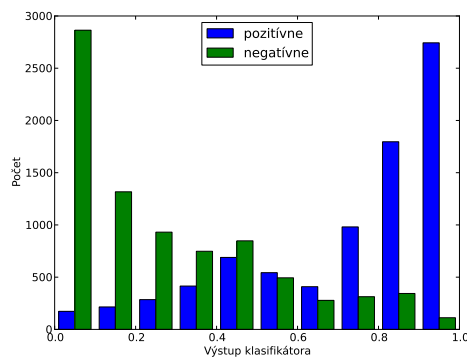
2.3.2 Úspešnosť klasifikátora

V tabulke 2.1 vidíme, že hoci typ C obsahuje nadmnožinu atribútov B, jeho úspešnosť je nižšia a tento typ nemá zmysel používať. Ďalej si môžeme všimnúť, že kombinácia typov A a B využila klady oboch a mierne ich vylepšila.

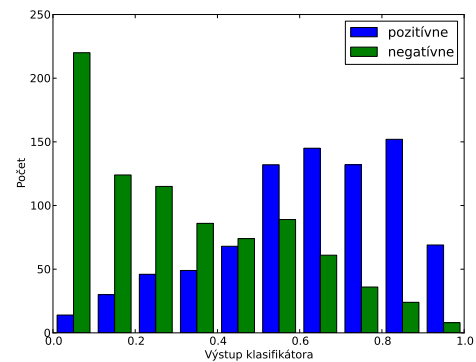


Obr. 2.4: Distribúcia výstupu z klasifikátora pri type dát D – modré sú pozitívne príklady a zelené su negatívne. Na x -ovej osi je výstup klasifikátora a na y -je počet inštancií, pre ktoré výstup z klasifikátora padol do daného chlievika

Úspešnosť klasifikátorov s rôznym typom dát sme skúmali aj podrobnejšie, pozerali sme sa aj na distribúciu výstupov klasifikátora. Silný klasifikátor má totiž distribúciu takú, že väčšina pozitívnych príkladov má vysoké výstupné hodnoty a väčšina negatívnych klasifikátorov má nízke výstupné hodnoty. Najlepšiu distribúciu výstupov sa nám podarilo dosiahnuť práve pri type D (obr. 2.4). Na obrázku 2.5 sú pre porovnanie najhoršie distribúcie z ostatných typov dát.



(a) Typ C: Match klasifikátor



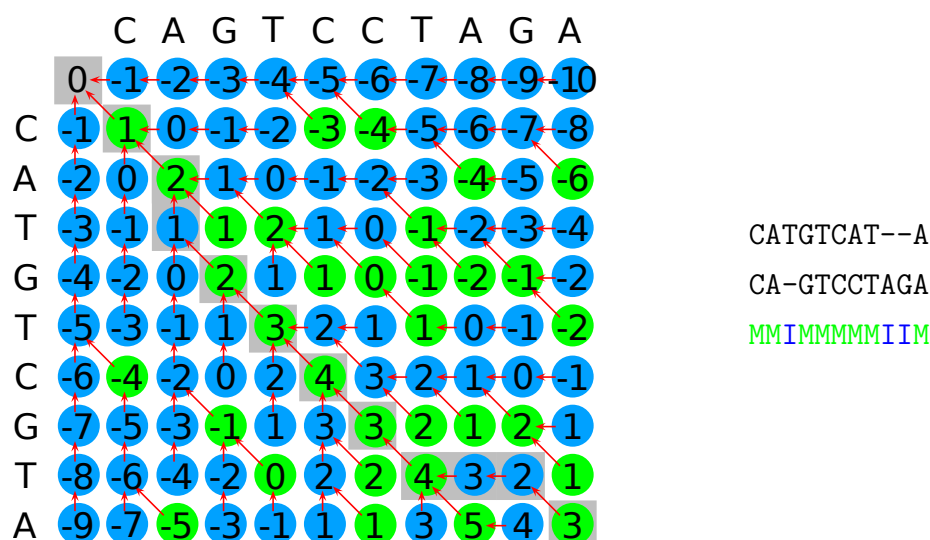
(b) Typ A: Indel klasifikátor

Obr. 2.5: Horšie distribúcie výstupov z klasifikátora – 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

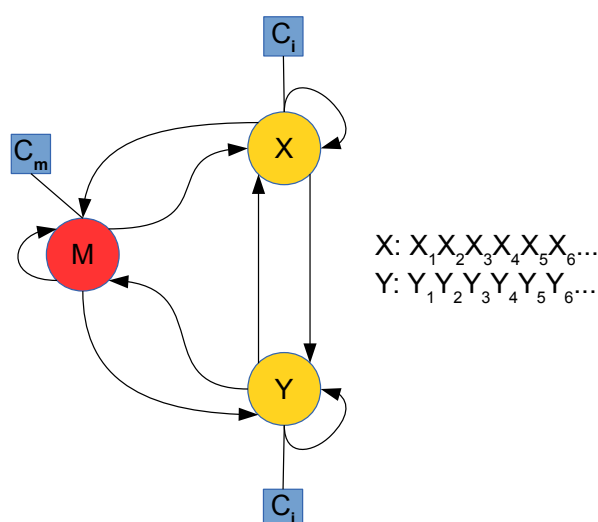
2.3.3 Nezávislosť typu dát na veľkosti okna

V tejto časti si ukážeme, že výber nášho typu dát nie je závislý na veľkosti okna. A teda môžeme najskôr vybrať typ dát a potom sa rozhodnúť, akú veľkosť okna použijeme.

Z tabuľky 2.2 vidíme, že klasifikátor s atribútmi typu D je, až na okno veľkosti tri, vždy najlepší. Rovnako zostalo zachované aj poradie ostatných typov dát. Z toho môžeme usúdiť, že pri voľbe typu dát nezáleží na veľkosti okna.



Obr. 3.1: Použité klasifikátory v klasifikátorovej páske pri zarovnávaní sekvencií. Pre jednoduchosť je použitá tabuľka z algoritmu Needleman-Wunch (kapitola 1.5.1) na globálne zarovnanie sekvencií. Viterbiho algoritmus používa tri tabuľky a cesta zarovnania vedie cez všetky tri. Zelenou farbou sú označené políčka, kde sa použije Match klasifikátor, modrou sú políčka, kde sa použije Indel klasifikátor. Šedou je vyznačené zarovnanie spolu s klasifikátorovou páskou pre zarovnanie. Vpravo je vypísané to isté zarovnanie spolu s klasifikátormi z klasifikátorovej pásky.



Obr. 3.2: Model s klasifikátorom ako emisiou

Vidíme teda, že zatiaľ čo

$$\sum_{x \in X, y \in Y} P(X = x \wedge Y = y) = 1,$$

pre

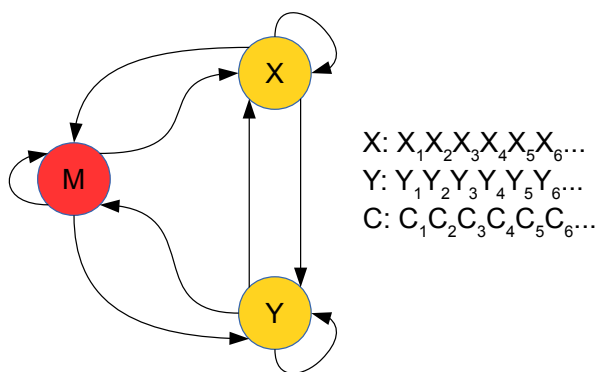
$$\sum_{x \in X, y \in Y} P(C = 1 | W = w)$$

takáto rovnosť neplatí. Viterbiho algoritmus však bude fungovať aj s hodnotami

$$P(C = 1 | W = w)$$

namiesto emisií a bude hľadať zarovnanie, ktoré bude maximalizovať takto definované skóre. O takomto modeli už však nemôžeme hovoriť ako o pravdepodobnostnom, ani ako o pHMM. Tento model je len inšpirovaný pHMM.

Keďže predošlý model nie je korektný pravdepodobnostný model, navrhli sme alternatívny model, *model s klasifikátorovou páskou (ďalej len model B)*, ktorý navyše modeluje aj výstup z klasifikátora, a teda je korektný pravdepodobnostný model. Ne-modelujeme len dvojicu sekvencií, ale aj sekvenciu výstupov klasifikátora vo forme pásky, ktorú sme si definovali na v sekcii 3. Pásku s výstupom z klasifikátora považujeme za akúsi pomôcku pre náš zarovnávač. Klasifikátor môže vrátiť ľubovoľnú



Obr. 3.3: Model s klasifikátorovou páskou

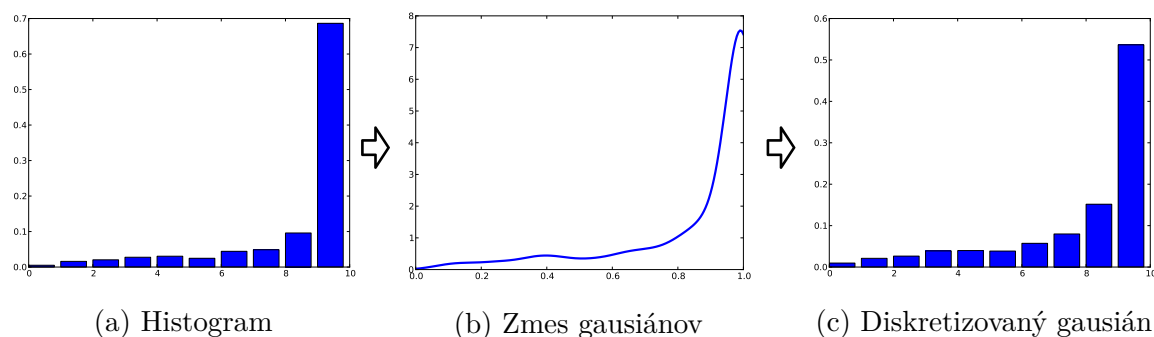
hodnotu z intervalu $\langle 0, 1 \rangle$. Klasické HMM robí iba s diskretnými hodnotami a keďže výstup z klasifikátora je spojitý, museli sme sa s týmto problémom vysporiadať. Preto sme navrhli dve varianty tohto modelu – diskretnú a spojitú.

V diskretnom modeli sme hodnoty klasifikátora rovnomerne rozdelili na b hodnôt a výstup sme zaokrúhlili na najbližšiu z nich. V našich modeloch sme zvolili $b = 10$.

Alternatíva k predošlej metóde je použiť spojitý HMM. Hlavný rozdiel medzi spojitými a diskretnými HMM je, že spojitý HMM nepočítajú s pravdepodobnosťou, ale s hustotou. Hodnota hustoty v danom bode na rozdiel od pravdepodobnosti môže byť aj väčšia ako jedna. Podľa [HHL89], jedným zo štandardných spôsobov reprezentácie hustoty v spojitých HMM je zmes gausiánov, takže využijeme interpoláciu takouto distribúciou, a to budeme brať ako hustotu výstupu klasifikátora (obr. 3.4b).

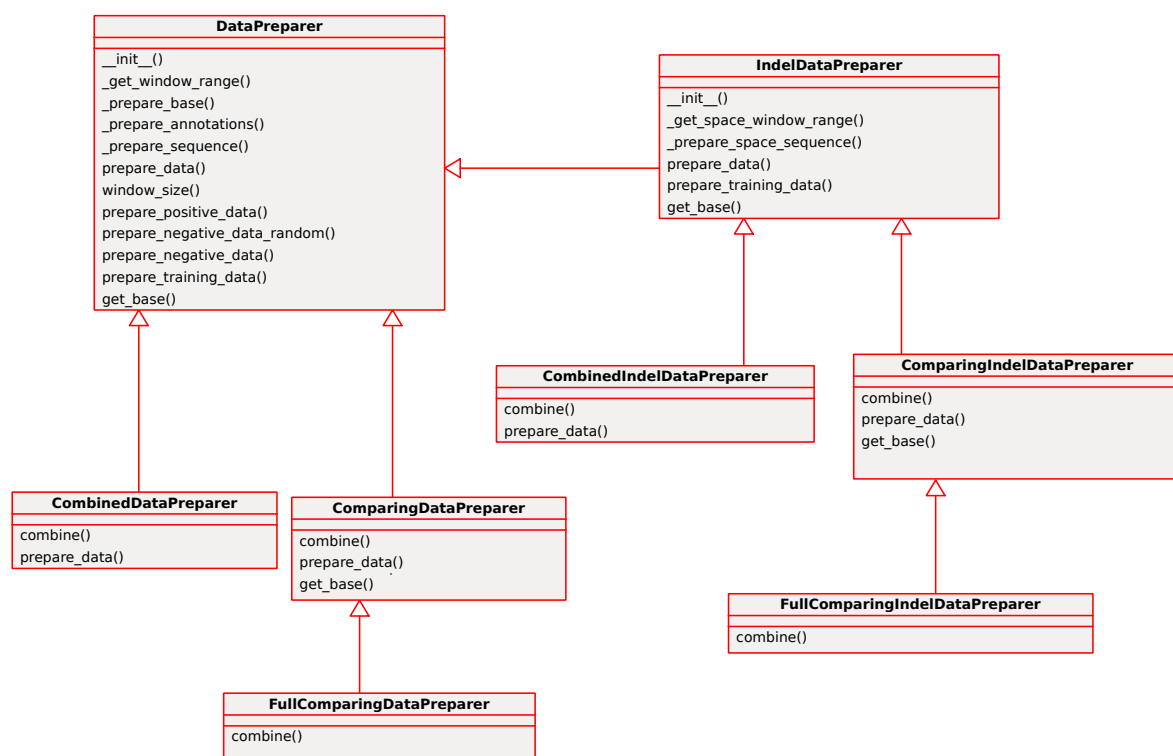
3.1.2 Diskretizácia skóre klasifikátorov

Pre diskretnú verziu modelu B sme sa zaoberali spôsobom diskretizácie. Diskretizovať hodnoty môžeme dvoma spôsobmi. Buď jednoducho spočítame histogram pre tréningové dáta, alebo interpolujeme vstupnú vzorku pomocou spojitej distribúcie, napr. pomocou zmesi gausiánov (tak ako v spojitý verzii) a potom toto rozdelenie diskretizujeme (obr. 3.4). Druhá spomenutá metóda má výhodu v tom, že vyhladí šum a doplní chýbajúce dáta, ale keďže je použitá aproximácia, zavádza určitú nepresnosť. V oboch prípadoch si musíme zvoliť počet košov b , do ktorých dáta rozdelíme. Koše sme rozdelili rovnomerne a na základe experimentov (tabuľka 3.1) sme si zvolili $b = 10$.



Obr. 3.4: Dve možnosti diskretizácie výstupu klasifikátora: buď použijeme histogram 3.4a, alebo dáta najskôr aproximujeme pomocou zmesi gausiánov 3.4b a následne zdiskretizujeme ako na obr. 3.4c.

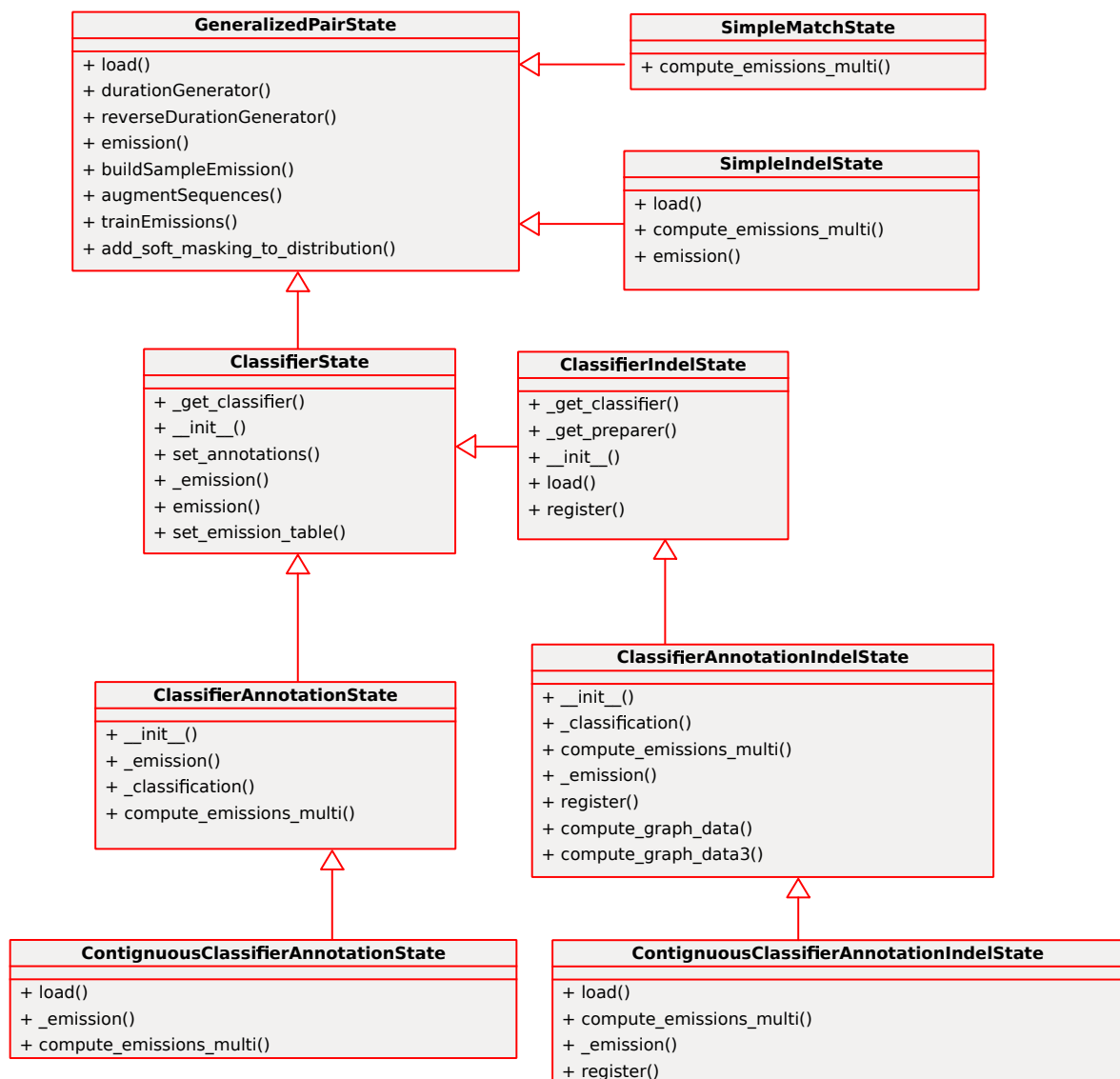
Ako vidíme z tabuľky 3.1, spojitá verzia modelu sa nám neosvedčila. Má totiž nedostatok, že distribučná funkcia nedosahuje maximum pri výstupe klasifikátora rovnom jedna, ale kúsok predtým, čo znamená istú penalizáciu v prípade, že si je klasifikátor príliš istý. Môžeme si tiež všimnúť prudký pokles úspešnosti pri diskretizácii pomocou



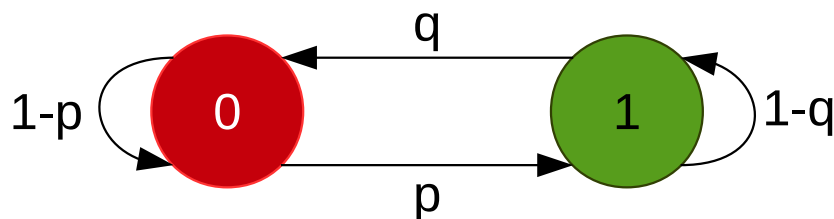
Obr. 4.1: Diagram tried pre DataPreparer-y

s hodnotou klasifikátora. Tieto triedy takisto obsahujú aj metódy na tréovanie emisií.

Od triedy *GeneralizedPairState* navyše dedia aj naše triedy pre referenčný model – *SimpleMatchState* a *SimpleIndelState* (obr. 4.2), ktoré triedu *GeneralizedPairState* rozširujú o metódy na tréovanie emisných pravdepodobností.



Obr. 4.2: Diagram tried pre stavy



Obr. 4.3: Markovova reťaz použitá na generovanie informácie o génoch. p je pravdepodobnosť, že začneme generovať gén, q je pravdepodobnosť, že prestaneme generovať gén.

do iného stavu, alebo ostaneme v tom istom. Rozhodnutie robíme pomocou *falošnej mince* (*biased coin*), kde hlava padne s danou pravdepodobnosťou, ktorú nastavíme ako parameter. V našom prípade je to p pre stav nula a q pre stav jedna.

Simulácia mutácie a delécie

Ak už máme vygenerovanú základnú sekvenciu, potom vyrobíme zmutovanú sekvenciu tak, že s určitou pravdepodobnosťou nahradíme bázu zo základnej sekvencie inou bázou. Pravdepodobnosť závisí aj od toho, či je na danej pozícii gén v oboch sekvenciách, v jednej, alebo v žiadnej. Na rozhodovanie máme pre každú možnosť jednu falošnú mincu a podľa toho, ktorá z možností nastala, hodíme si príslušnou mincou.

Deléciu simulujeme opäť pomocou podobnej Markovovskej reťaze, pretože počas evolúcie majú tendenciu vypadávať súvislé úseky. Stav nula bude, že nemažeme a stav jedna, že mažeme, teda nahradzujeme danú bázu znakom '- '.

Využitie

Simulátor je prvá vec, ktorú sme implementovali a slúžil na väčšinu našich experimentov. Simulované dáta majú totiž oproti biologickým niekoľko výhod. Prvou je, že poznáme správnu odpoveď, a teda môžeme objektívne zhodnotiť úspešnosť našich modelov. Druhou je možnosť zvoliť si parametre, ktoré menia závislosť mutácií na anotáciách a testovať správanie modelov, ak zvýšime túto závislosť.

4.4.1 Konfigurácia a Formáty súborov

Konfigurácia našich modelov sa nachádza v dvoch súboroch:

- `classifier_alignment/constants.py` – tu sa dá zmeniť veľkosť okna a či sú povolené anotácie
- `classifier_alignment/config.py` – tu sa dá zmeniť klasifikátor, datapreparer a či sa má použiť ten istý klasifikátor aj na Match aj na Inzert stav alebo nie. V tomto súbore sú preddefinované klasifikátory a datapreparer-y takže stačí zmeniť index označujúci, ktorý z nich sa má použiť.

Súbory zarovnania sú vo formáte FASTA¹, súbory anotácií sú vo formáte BED².

Súbory modelov sú vo formáte JSON³, nasledujúceho tvaru:

```
1 {
2   "model": {
3     "states": [
4       {
5         "name": "Match",
6         "durations": [[[1, 1], 1.0]],
7         "startprob": 0.33,
8         "endprob": 1.0,
9         "emission": [],
10        "onechar": "M",
11        "__name__": "SimpleMatchState"
12      },
13      {
14        "name": "InsertX",
15        "durations": [[[1, 0], 1.0]],
16        "startprob": 0.33,
17        "endprob": 1.0,
18        "emission": [],
19        "onechar": "X",
20        "__name__": "SimpleIndelState"
21      },
22    ]
23  }
```

¹http://en.wikipedia.org/wiki/FASTA_format

²<http://genome.ucsc.edu/FAQ/FAQformat.html#format1>

³JavaScript object notation

```

23     "name": "InsertY",
24     "durations": [[[0, 1], 1.0]],
25     "startprob": 0.33,
26     "endprob": 1.0,
27     "emission": [],
28     "onechar": "Y",
29     "__name__": "SimpleIndelState"
30   }
31 ],
32   "__name__": "GeneralizedPairHMM",
33   "transitions": []
34 }
35 }

```

Emisné a prechodové tabuľky sa vyplnia pri tréňovaní modelu. Zaujímavé sú pre nás len položky `__name__` v jednotlivých stavoch. Tam treba vložiť meno príslušnej triedy pre stav.

Súbor pre konfiguráciu anotácií je JSON nasledujúceho tvaru:

```

1 {
2   "__name__": "Annotations",
3   "annotations": ["gene"],
4   "sequences": [
5     {
6       "name": "sequence1", "annotations": [
7         {
8           "id": "gene",
9           "file": "data/sequences/simulated/
              simulated_alignment_sequence1_gene.bed"
10          "offset": 47
11        }
12      ]
13    },
14    {
15      "name": "sequence2", "annotations": [
16        {
17          "id": "gene",

```

```
18         "file": "data/sequences/simulated/  
19             simulated_alignment_sequence2_gene.bed"  
20     "offset": 47  
21     }  
22 ]  
23 },  
24 {  
25     "name": "sequence3", "annotations": [  
26     {  
27         "id": "gene",  
28         "file": "data/sequences/simulated/  
29             simulated_alignment_sequence3_gene.bed"  
30         "offset": 47  
31     }  
32 ]  
33 }
```

Do `annotations` treba dať zoznam anotácií ktoré sa majú použiť a do `sequences` treba dať zoznam sekvencií – meno musí byť rovnaké ako vo FASTA súbore. Pre každú sekvenciu treba vyplniť v `annotations` pre každú anotáciu (id musí byť rovnaké ako v zozname anotácií) zdrojový súbor, odkiaľ sa má daná anotácia načítať. Navyše je možné špecifikovať `offset` v prípade, že sekvencia, ktorú chceme zarovnať, nezačína v genóme na pozícii 0 a anotácie sú číslované podľa pozícií v genóme.

4.5 Rozšírenie programu

Program je možné ľahko rozšíriť rôznymi spôsobmi. Je možné pridať:

- novú triedu pre stav
- nový DataPreparer
- nový klasifikátor