Univerzita Komenského, Bratislava
Fakulta Matematiky, Fyziky a Informatiky

Kompresia genómu za pomoci skrytých Markovovských modelov

Bakalárska práca

2012

Jakub Kováč

Univerzita Komenského, Bratislava
Fakulta Matematiky, Fyziky a Informatiky

Kompresia genómu za pomoci skrytých Markovovských modelov

Bakalárska práca

Študijný program:	Informatika
Študijný odbor:	2508 Informatika
Školiace pracovisko:	Katedra Informatiky
Školiteľ:	RNDr. Tomáš Vinař, PhD.

Bratislava, 2012
Jakub Kováč

Ďakujem svojmu školiteľovi Tomášovi Vinařovi a taktiež Broni Brejovej za konštruktívnu kritiku za cenné rady a pomoc pri vypracovávaní mojej práce. Tiež priateľom a rodine za pomoc, podporu a v neposlednom rade všetkým ktorí sa za mňa modlili. Na záver chcem poďakovať Martinovi Králikovi a Mišovi Hozzovi za ušetrený čas a nervy pri práci s texom.

Jakub Kováč

Abstrakt

Kompresia vznikla ako odpoveď na efektívne uchovávanie dát a tiež na urýchlenie prenosu informácie na sieťach s pomalým tokom dát. Vzniklo viacero algoritmov bezstratovej kompresie dát, či už všeobecných alebo určených pre konkrétny typ dát.

S rozvojom genetiky sa objavil úplne nový typ dát hovoriaci o poradí báz v kyseline deoxyribonukleovej (sekvencie DNA). Tieto dáta nie je možné komprimovať bežne používanými všeobecnými algoritmami. Naše riešenie problému kompresie DNA je založené na použití aritmetického kódovania so skrytým Markovovským modelom (HMM) ako adaptívnym modelom zdroja.

Po implementovaní programu sme na sekvencii DNA natrénovali niekoľko HMM, ktorými sme modelovali rôzne vlastnosti sekvencie DNA. Následne sme týmito modelmi komprimovali inú sekvenciu DNA. Na rozdiel od všeobecných komprimačných algoritmov, pri ktorých mala komprimovaná sekvencia väčšiu veľkosť ako nekomprimovaná sa nám podarilo dosiahnuť nepatrné zlepšenie aj oproti priamočiaremu zakódovaniu každej bázy dvomi bitmi.

Otvorenou otázkou ostáva hľadanie vhodnejších HMM sekvencie DNA a aký je najlepší kopresný pomer dosiahnuteľný touto metódou. Tu treba mať na zreteli, že zväčšenie a spresnenie použitého HMM spomaľuje kompresiu a následnú dekompresiu DNA.

Kľúčové slová: kompresia, DNA, HMM, aritmetické kódovanie, genóm

Abstract

Compression was created for effective data storage and also to accelerate the transfer of information on networks with low data rates. Many lossless data compression algorithms have been created, whether general or designated for a specific type of data.
The development of genetic and biochemistry have brought completely new type of data. It is data about the order of nucleotids in the molecule of deoxyribonucleic acid (DNA sequence). General compress algorithms fails in compression of this data. Our solution to the problem of DNA compression is based on the use of arithmetic coding with Hidden Markov Models (HMM) as an adaptive model of the source.
After the implementation of our program we suugested and treined several HMM for a DNA sequence. This HMM‘s modele different properties of DNA sequences. Then we use this models for compression of another DNA sequence. Unlike the general compression algorithm, which compressed sequence used more space tnan uncompressed we have achieved a slight improvement compared to basic encoding of one nucleotid by two bits.
An open question is searching more appropriate HMM for DNA sequences and what is the best kopression ratio achievable with this method. It should be borne in mind that the extension and refinement used HMM slow compression and following decompression of DNA.

Key words: compression, DNA, genome, HMM, aritmetic encoding

Čestne prehlasujem, že som túto bakalársku prácu vypracoval samostatne s použitím citovaných zdrojov.

.....

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Jakub Kováč
Študijný program:	informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor:	9.2.1. informatika
Typ záverečnej práce:	bakalárska
Jazyk záverečnej práce:	slovenský

Názov: Kompresia genómov za pomoci skrytých Markovových modelov

Cieľ: Skryté Markovove modely (HMM) sa často používajú na modelovanie rôznych častí DNA. V tejto práci bude cieľom preskúmať, ako by sa dali použiť na kompresiu jedného alebo viacerých príbuzných genómov.

Vedúci: Mgr. Tomáš Vinař, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky

Dátum zadania: 14.10.2011

Dátum schválenia: 26.10.2011 doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

Obr. 1: Text pod obrazkom

Obsah

Úvod

Všetky metódy kompresie dát vieme rozdeliť do dvoch skupín: stratové a bezstratové. Prvá skupina sa používa v prípadoch, keď nie je nutná presná obnova komprimovaných dát napr. pri súboroch s hudbou alebo videom. Druhá skupina zahŕňa metódy kompresie umožňujúce úplnú rekonštrukciu vstupných dát.

Vzniklo viacero metód a techník bezstratovej kompresie dát a niektoré z nich boli vyvinuté špeciálne pre dáta z konkrétneho zdroja. Dobré výsledky sa dosahujú napríklad pri kompresii ľudských textov, obzvlášť ak ich vydali inštitúcie Európskej únie. Taktiež existujú spôsoby efektívnej kompresie dát pochádzajúcich z rôznych fyzikálnych meraní.

Spolu s rozvojom genetiky vznikol úplne nový druh dát. Sú to dáta o zložení molekuly kyseliny deoxyribonukleovej (DNA). Molekula DNA je v bunke nositeľom drvivej väčšiny potrebnej informácie. Táto táto informácia je v nej uložená v poradí jej častí zvaných bázy. Tieto bázy sú štyri adenín (A), cytozín (C), guanín (G) a tymín (T). Informácia o ich poradí, zvaná sekvencia DNA, je dlhé slovo na abecedou $\{A, C, G, T\}$.

Keďže pri kompresii sekvencie DNA všeobecné bezstratové kompresné algoritmi zlyhali a priestor potrebný pre uloženie nimi skomprimovanej sekvencie bol väčší ako neskomprimovanej, skúma sa kompresia tohto typu dát zvlášť. Vzniklo viacero riešení problému kompresie sekvencie DNA s rôznou úspešnosťou.

Náš prístup spočíva v kompresii sekvencie DNA pomocou aritmetického kódovania s adaptívnym modelom. Ako typ modelu sme sa rozhodli vybrať skrytý Markovovský model (HMM), pretože jednoduché pravdepodobnstné modely nedokážu dostatočne vystihnúť sekvenciu DNA. HMM je stavový pravdepodobnostný model, ktorým dokážeme modelovať náhodný proces emitujúci vo svojom udalosť a následne prachádzajúci do ďalšieho stavu; toto všetko s určenou pravdepodobnosťou. Pritom poznáme postupnosť udalostí, ktoré model emitoval a postupnosť stavov v ktorých pritom bol nám ostáva skrytá. Špecifikom tejto sekvencie totiž je, že celková frekvencia báz je veľmi podobná. Avšak molekula DNA nie je homogénna a skladá sa z mnohých typov úsekov

s navzájom rôznymi frekvenciami jednotlivých báz. Tieto úseky modelujeme stavmi HMM, pričom ich zoskupujeme do typov podľa podobnosti v rôznych vlastnostiach.

Algoritmus Huffmanovho kódovania sa, napriek vynaloženému úsiliu na jeho implementáciu a strávenému času, ukázal ako neperspektívny, aj keď z programátorského hľadiska vďaka svojej zložitosti zaujímavý, preto sa mu v našej práci nevenujeme.

Naše výsledky zatiaľ nedosahujú úroveň kompresného pomeru algoritmov špecializovaných na DNA, ale dúfame, žeje potenciál ho zlepšiť použitím stále vhodnejších skrytých Markovovských modelov.

V prvej kapitole našej práce sa venujeme kompresii vo všeobecnosti a následne kompresii DNA, ako poslednej sa v nej venujeme myšlienke aritmetického kódovania, ktoré je podrobne opísané v štvrtej kapitole v časti o algoritme.

V druhej kapitole sa nachádzajú definície skrytého Markovovského modelu. Taktiež sa venujeme modifikácii HMM ako HMM vyššieho rádu a samotnej tvorbe HMM, teda návrhu jeho štruktúry a trénovaniu pravdepodobností.

Tretia kapitola našej práce je o vlastnostiach sekvencie DNA a modeloch, ktorými sme sa ich sanžili modelovať. Tiež spomíname dosiahnuté výsledky.

Štvrtá kapitola sa týka algoritmov a špecifických prístupov (napr. reprezentácia malých čísel ich logaritmom) použitých v kompresnom programe.

Kapitola 1

Kompresia dát

Kompresia je proces úpravy dát tak, aby zaberali čo najmenej miesta. Pri tomto procese z dát odstraňujeme nadbytočnú informáciu. Opačný proces, pri ktorom z komprimovaných dát spätne vytvárame pôvodnú informáciu sa nazýva dekompresia. Kompresné metódy sa delia do dvoch základných skupín podľa toho, či sa pri dekompresii dajú obnoviť dáta v plnom rozsahu alebo nie. Prvá skupina sa nazýva bezstratové algoritmy a druhádruhá skupina sa nazýva stratové algoritmy. Stratové algoritmy sa používajú hlavne pri kompresii videa a hudby, kde nie je potrebná presná rekonštrukcia pôvodného súboru. Kompresia sa vo všeobecnosti používa jednak pri archivovaní informácií a jednak pri prenose dát cez pomalé siete, kde chceme ich objem čo najviac zmenšiť a tým urýchliť dátový prenos.

1.1 Bezstratová kompresia

Ako uvádza [HH94] v prvej skupine – bezstratovej kompresii sa vyvinuli dve teórie. Prvú publikoval Claude Shannon v roku 1948 [Sha48] a zodpovedajú jej v podstate všetky súčasné bezstratové kompresné algoritmy. Druhú teóriu objavill v roku 1974 Klaus Holtz a v krátkom čase si patentoval na nej založenú kompresnú aplikáciu (Patent Holtz 4 366 551). Kým Shannonova teória sa pozerá na reťazec bitov ako na odpovede áno-nie na hypotetické otázky, ktorých cieľom je odstrániť „neistotu“ u prijímateľa, tak Holtzova teória, zvaná „Autosophy“ je založená na učení a úlohou informácie podľa nej nie je „odstrániť neistotu“, ale má „vytvoriť vedomosť“. V oboch teóriách majú odosielateľ aj prijímateľ nejakú predchádzajúcu vedomosť o charaktere informácie a dôležitá je len informácia o tom, čo ešte prijímateľ nevie. Všetka informácia, ktorú vie prijímateľ zistiť bez toho, aby o nej dostal správu od odosielateľa, je nadbytočná.

Pri kompresii sa snažíme z dát odstrániť čo najviac nadbytočnej informácie, pretože si ju z nich vieme zrekonštruovať a nepotrebujeme ju uchovávať. Ide najmä o rôzne

pravidelnosti v komprimovaných dátach, ako napríklad opakujúce sa podreťazce, či rôzna frekvencia znakov na vstupe.

1.1.1 Entropická kompresia

Entropická kompresia využíva rôzne pravdepodobnosti výskytu jednotlivých symbolov a reťazcov na vstupe. Medzi najznámejšie algoritmy z tejto triedy patrí Huffmanov kód, aritmetické kódovanie a Shannonov-Fannov kód (tiež nazývaný len Fannov kód) [Say96]. Keďže pre každú sekvenciu znakov vytvárame reťazec bitov, ktorý ich kóduje, môžeme sa na ne aj vo všeobecnosti pozrieť ako na listy binárneho stromu, kde jednotlivé bity určujú cestu z koreňa do listov reprezentujúcich jednotlivé sekvencie. Tento strom budeme nazývať kódovací strom. V tomto prípade sa môžeme aj na aritmetické kódovanie pozrieť ako na algoritmus kódujúci cestu v nejakom kódovacom strome. Medzi aritmetickým kódovaním na jednej strane a Huffmanovým a Fannovým na strane druhej, však existuje niekoľko rozdielov. Algoritmus Huffmanovho kódovania buduje tento strom odspodu, keď dvom vrcholom s najmenšou váhou z tých, ktoré ešte nemajú otca, pridá otca, ktorého váhou bude súčet váh týchto vrcholov. Toto opakuje dovtedy, kým neostane len jeden vrchol bez otca a ten sa stane koreňom kódovacieho stromu. Naproti tomu Fannov kód buduje kódovací strom odvrchu, keď rekurzívne delí množinu sekvencií na dve polovice s približne rovnakou pravdepodobnosťou a jednu priradí ľavému a druhú pravému synovi aktuálneho vrchola. Oba tieto algoritmy teda budujú kódový strom pre celú množinu možných sekvencií

Huffmanove kódovanie a Fannovo kódovanie budujú celý kódovací strom a preto sa, kvôli obmedzenej pamäti, používajú ako blokové. Blokové algoritmy rozdelia komprimované dáta na kratšie podreťazce – bloky a tie kódujú samostatne. Bloky môžu mať dĺžku od jedného písmenka po kratší reťazec. Tieto algoritmy potom budujú kódový strom pre blok danej dĺžky. Naproti tomu aritmetické kódovanie sa pozerá akoby len na cestu ku komprimovanej sekvencii a jej blízke okolie, teda negeneruje celý kódovací strom, a preto môže komprimovať vstupné dáta ako jeden celok a nemusí ich deliť na bloky. Táto odlišnosť aritmetického kódovania nám umožňuje kódovať znaky alebo ich reťazce neceločíselným počtom bitov. Pri zvyšných dvoch kódovaniach sme nútený priestor spotrebovaný na blok zaokrúhliť smerom hore na celé bity.

Entropické kódy sa väčšinou používajú buď spolu s modelmi, ktoré dokážu predpovedať pravdepodobnosť jednotlivých symbolov na vstupe alebo spolu s rôznymi spôsobmi predspracovania, ktorých cieľom je zvýšiť konečný kompresný pomer a predpripraviť dáta pre použitie entropických komprimačných techník.

Pokiaľ očakávame, že dáta na vstupe budú pozostávať z pomerne dlhých reťazcov opakujúcich sa písmen, používa sa väčšinou takzvané run-length kódovanie. V tomto

prípade sa o dátach uchová len informácia aké písmeno a koľko krát za sebou sa opakuje.

1.1.2 Všeobecné bezstratové kompresné algoritmy

V prípadoch, keď je potreba uložiť veľké množstvo dát rovnakého druhu je hlavnou stratégiou vývoj algoritmu optimalizovaného pre tento typ dát. Vo všeobecnosti nie je tento prístup výhodný, pretože náklady na vývoj špecializovaného algoritmu by prevýšili výhody získané zmenšením potrebného priestoru a vzniklo by obrovské množstvo kompresných algoritmov. Keby sme však nevedeli nič o dátach, nedokázali by sme identifikovať nadbytočnú informáciu. Všeobecne prijímaným kompromisom je rozdelenie dát do tried a definovanie zdroja pre každú z nich. V tomto prípade predpokladáme, že dáta sú produkované zdrojom z niektorej triedy a použijeme kompresný algoritmus pre danú triedu.

Medzi najstaršie zo všeobecných bezstratových kompresných algoritmov nich patria algoritmy, ktoré vypracovali Ziv and Lempel v rokoch 1977-1978 LZ77 [ZL77] a LZ78 [ZL78]. LZ77 a LZ78 sú slovníkové algoritmy, ktoré nahrádzajú výskyt slova ukazateľom na jeho výskyt v slovníku. Okno je podreťazec reťazca danej dĺžky a polohy. LZ77 používa dve okná, v prvom (ozn. SB) je posledná zakódovaná časť sekvencie a v druhom (ozn. LAB) je časť sekvencie, ktorú ide zakódovať.

Algoritmus "prediction by partial matching"(PPM) [CW84] na základe predošlého kontextu predpovedá pravdepodobnosť s akou sa ten-ktorý znak vyskytne na vstupe. Frekvencia symbolov je ukladaná zvlášť pre každý kontext, pričom ako kontext sa berie posledných n znakov. Využíva na to skryté Markovovské modely s adaptívnymi pravdepodobnosťami emisií. Aj keď majú PPM z triedy všeobecných bezstratových algoritmov aj dnes najlepšie kompresné pomery, ich hlavnými nevýhodami ostáva pomalá rýchlosť a veľká spotreba pamäte. Napriek tomu sa používajú v praxi.

Ďalšia štatistická kompresná metóda "dynamic Markov coder"(DCM) [CH87] predpokladá, že komprimované dáta sú výstupom dátovej triedy Markovovských zdrojov. Počas kompresie sa snaží čo najlepšie tento zdroj vystihnúť stále presnejším určením pravdepodobností symbolov na základe už prečítanej sekvencie. Počas kompresie sa použítý dynamický skrytý Markovovský model zväčšuje, aby bol odhad pravdepodobností jednotlivých znakov čo najpresnejší, čím sa zvyšujú nároky na pamäť. Tieto pravdepodobnosti potom algoritmus kóduje používajúc aritmetický kóder. Kompresné pomery čas behu DCM a PPM sú porovnateľné, avšak [Deo03] uvádza, že v oblasti PPM nastal značný pokrok, kým vývoj DCM stagnuje.

Willems et al. navrhol v roku 1995 "context tree weighting"(CTW) algoritmus [WST95]. Autor zavádza koncept triedy zdrojov context tree a vzhľadom na ňu odhaduje pravdepodobnosť výskytu symbolov na vstupe. Pri tejto metóde slúžia vypočítané

pravdepodobnosti, podobne ako pri iných metódach, ako vstup pre aritmetický kóder. Výhodou tejto metódy je, že napriek ešte len rozbiehajúcemu sa výzkumu dosahuje kompresný pomer blízky PPM, sčasne pri nízkej rýchlosti behu.

1.2 Kompresia DNA

Kyselina deoxyribonukleová (DNA) má v bunkách funkciu nosiča genetickej informácie. Táto informácia je uložená v poradí nukleotidov. Nukleotid je základná stavebná jednotka DNA. Úsek molekuly DNA voláme aj sekvencia DNA. Keďže tieto nukleotidy sú štyri: adenín, cytozín, guanín a tymín, môžeme sa na sekvenciu DNA pozerať ako na slovo nad štvorpísmenovou abecedou {A,C,G,T}.

V súčasnej dobe narastá množstvo údajov získaných sekvenovaním DNA a narastá dôležitosť otázky ich efektívneho uchovávania. DNA predstavuje štruktúrovanú informáciu, pri ktorej záleží na výskyte a poradí jednotlivých písmen, podobne ako bežný text. Pomocou bežných komprimačných nástrojov sa však nedá komprimovať, lebo frekvencie jednotlivých písmen sú veľmi podobné a po dvojiciach rovnaké. Bežne používané kompresné algoritmy dosahujú dokonca horší kompresný pomer ako keď zakódujeme jedno písmeno dvomi bitmi.

Najschodnejšou cestou preto ostáva hľadanie nadbytočnej informácie špecifickej pre DNA. Pomerne časté sú napríklad repetície, ktoré využívajú algoritmy založené na hľadaní podobných podreťazcov. Medzi kompresiou DNA a pochopením informácie, ktorú nesie je priama súvislosť. Čím viac vieme o jej štruktúre a o informácii, ktorú nesie, tým lepšie ju vieme skomprimovať. Na druhej strane, ak nájdeme nový spôsob ako ju ešte efektívnejšie komprimovať, môžeme sa dozvedieť niečo nové aj o jej štruktúre.

1.2.1 Rozdelenie kompresných metód pre DNA

V tejto oblasti prebieha od jej vzniku intenzívny vývoj. Postupne bolo vyvinutých množstvo algoritmov a [GSU09] uvádza štyri základné metódy kompresie DNA: substitučne-štatistická, transformačná, založená na gramatike a tabuľková.

Substitučne-štatistická metóda je kombináciou dvoch techník substitučnej a štatistickej. Pri kompresii sa sekvencia rozdelí na podreťazce a časť z nich sa skomprimuje jednou a časť druhou technikou. Táto paradigma sa začala používať v algoritmoch Biokompress 1 [GT93] a 2 [GT94]. Substitučnou technikou najprv zakóduje opakujúce sa reťazce a palindrómy, pričom nahradí opkakujúce sa výskyty podreťazca odkazom na jeho prvý výskyt. Na text, kde sa nenachádzajú žiadne významné opakovania ani palindrómy, použije štatistickú techniku a zakóduje ich aritmetickým kódovaním. Použitie substitučnej techniky je obmedzené množstvom zdrojov času a pamäte. Neskôr vzniklo

veľa iných algoritmov, ktoré rôzne vyvažovali čas a kompresný pomer. V roku 2007 sa objavil prvý algoritmus založený na čisto štatistickej metóde XM [CDAM07]. Používa aritmetické kódovanie a frekvenciu symbolov počíta z už spracovanej časti sekvencie. Autorom sa pomocu neho podarilo dosiahnuť kompresný pomer 1,6940 bit/báza. Tým prekonal všetky dovtedy publikované algoritmy na kompresiu DNA a dokonca aj všetky algoritmy na komresiu bielkovinových sekvencií. Popritom dokázal zachovať praktickú dobu behu.

Transformačná kompresná metóda sa zakladá na známej Burrows-Wheeler transformácii [BWBW94], tiež zvanej block-sorting. Podstatou tejto transformácie je, že sa z transformovávaného bloku dát vytvoria všetky rotácie, utriedia sa abecedne a na výstup sa dajú posledné znaky z týchto rotácií a umiestnenie pôvodného bloku v tomto utriedení. Na tejto trandformácii sú založené len dve navzájom variantné metódy špecializované na biologickú kompresiu. Táto transformácia je ukážkovým príkladom, ako nám môže pomôcť kompresia lepšie spoznať charakter dát, keď ukázala dovtedy neznámu redundanciu. Napríklad pri bielkovinových sekvenciách sa pomocou nej podarilo objaviť vysoké množstvo opakovaní podreťazcov, o ktorom sa predtým nevedelo.

Kompresná metóda založená na gramatikách je založená na bezkontextovej triede gramatík. Pod túto metódu spadajú aj niektoré algoritmy komprimujúce DNA aj s anotáciou. Pri tejto metóde sa hľadá gramatika najlepšie vystihujúca danú sekvenciu DNA alebo skupinu sekvencií a následne sa kóduje odvodenie v nej a nakoniec sa kompresia aplikuje aj na výsledný reťazec bitov. Napriek úspešnosti tejto metódy na iných typoch dát došiel [CL04] k záveru, že táto metóda nie je vhodná na kompresiu DNA.

Poslednou spomínanou kompresnou metódou je tabuľková metóda [BFG]. Tabuľky sú súbory záznamov rovnakej dĺžky pochádzajúcich z jedného zdroja a preto majú veľmi vysokú mieru redudancie. Môže ísť napríklad o záznamy o hovoroch, meteorologické údaje alebo sekvencie toho istého chromozómu od rôznych jedincov. Tabuľka sa najprv rozdelí na partície a tie sa potom komprimujú zvlášť. Táto metóda je veľmi efektívna pri kompresii súboru podobných sekvencií. Bola s vynikajúcimi výsledkami použitá na rôznych typoch tabuľkových dát.

1.3 Aritmetické kódovanie

Aritmetické kódovanie je bezstratový entropický komprimačný algoritmus, ktorý kóduje vstupnú sekvenciu znakov na výstupnú sekvenciu znakov (zväčša bitov) vyvužívajúc prevažne základné aritmetické operácie sčítanie (odčítanie) a násobenie (delenie).

Hlavnou myšlienkou je, že algoritmus priradí každej sekvencii podinterval intervalu

$< 0, 1)$ a ten následne zakóduje. Pričom platí, že interval sekvencie je podintervalom intervalov všetkých prefixov danej sekvencie. Toto nám umožňuje kódovať sekvenciu priebežne popri jej čítaní, takže si ju nemusíme pamätať a pamäťová náročnosť je vzhľadom na dĺžku sekvencie konštantná. Prázdnej sekvencii prislúcha interval $< 0, 1)$. Taktiež platí, že interval každej sekvencie je zjednotenie intervalov všetkých sekvencií, ktorým je ona prefixom a intervaly dvoch frekvencií, kde ani jedna nie je prefixom druhej sú disjunktné. Tento interval je priamo úmerný pravdepodobnosti sekvencie.

Interval sa zakóduje ľubovoľným číslom z neho, keďže pri kompresii sa šetrí miestom, používa sa číslo s čo nakratším zápisom. Súčasťou algoritmu je aj model zdroja, pomocou ktorého počítame pravdepodobnosti sekvencií. Tento algoritmus je optimálny ak kóduje zdroj s konštantnými nezávislými frekvenciami znakov.

Kapitola 2

Skryté Markovovské modely (HMM)

Markovovské modely sú stavové pravdepodobnostné modely široko používané nielen v bioinformatike ale napríklad aj pri rozpoznávaní reči či písma.

HMM sa používa na modelujovanie procesu, ktorý je v nám neznámom stave. V jednom kroku náhodne v tomto stave emituje znak (udalosť) a prejde do ďalšieho stavu (môže prejsť aj do toho istého), pričom sú dané pravdepodobnosti emisie jednotlivých znakov a prechodov do jednotlivých stavov.

Pomocou HMM sa snažíme o modelovanom procese po každom jeho kroku čo najpresnejšie určiť s akými pravdepodobnosťami je v danom stave. Poznáme pravdepodobnosti s ktorými bol tento proces v jednotlivých stavoch a znak, ktorý v nich emitoval. Na základe toho vypočítame podmienené pravdepodobnosti, že proces, keď emitoval daný znak, bol v tom-ktorom stave. Z týchto podmienených pravdepodobností a tabuľky s pravdepodobnosťami prechodov zo stavu do stavu vypočítame s akými pravdepodobnosťami je modelovaný proces po tomto kroku v jednotlivých stavoch.

Týmto spôsobom modelujeme napríklad ľudskú reč, keď sa snažíme zo zvuku zistiť, ktorú hlásku človek vyslovuje, teda v akom je stave. V tomto prípade nás zaujíma najpravdepodobnejšia postupnosť stavov k danej postupnosti udalostí (zvukov).

V našej práci modeluje pomocou HMM pravdepodobnosť konkrétnej postupnosti báz v molekule kyseliny deoxyribonukleovej (sekvencie DNA). O sekvencii DNA predpokladáme, že pravdepodobnosť výskytu jednotlivých báz sa v nej mení v závislosti od typu predchádzajúcich báz. Túto zmenu modelujeme zmenou pravdepodobností stavov. V každom kroku počítame pravdepodobnosť danej bázy a celkovú pravdepodobnosť sekvencie počítame ako súčin pravdepodobností, že za sebou budú nesledovať jenotlivé znaky sekvencie.

Na základe vypočítanej pravdepodobnosti sekvencie DNA sa ju následne snažíme zakódovať tak, aby sme na pravdepodobnejšiu sekvenciu použili primerane menej bitov

ako na menej pravdepodobnú.

V tejto kapitole si zadefinujeme skryté Markovovské modely, pričom pravdepodobnosti definujeme ako zobrazenia do množiny reálnych čísel, aby sa nám s nimi lepšie pracovalo.

Pravdepodobnosti s ktorými je modelovaný proces v jednotlivých satvoch sa nazývajú distribúcia stavov. Pravdepodobnosti emisie znakov budeme nazývať emisné pravdepodobnosti a pravdepodobnosti prechodu zo stavu do stavu budeme nazývať prechodové pravdepodobnosti.

Pri našich výpočtoch budeme využívať aj markovovské modely vyššieho rádu. Tieto sa od vyššie definovaného modelu líšia tým, že pravdepodobnosti emisií jednotlivých znakov v danom stave závisia od predtým vypísaných znakov. Rád stavu je počet predtým vypísaných znakov od ktorých závisia emisné pravdepodobnosti z tohto stavu. HMM vyššieho rádu nám umožňujú upresniť pravdepodobnosti pravdepodobnosti emisií znakov bez nutnosti pridať ďalšie stavy.

Pravdepodobnosti emisií znakov v danom stave a pravdepodobnosti prechodov zo stavu do stavu sa nemenia. Rovnako sú konštantné aj množina stavov a množina znakov (abeceda).

2.1 Definície

V našej práci používame skrytý Markovovský model na modelovanie pravdepodobností jednotlivých báz v genóme.

Definícia 1
Skrytý Markovovský model (HMM) je 5-ica $\{\Sigma, S, M, E, A\}$, kde Σ je abeceda výstupných symbolov, S je množina stavov, M je zobrazenie $S \times S \to R$,
E je zobrazenie $S \times \Sigma \to R$ a A je zobrazenie $S \to R$,
Platí:
$\forall g \in S : \sum_{i=1}^{|S|} M(g, s_i) = 1$, kde $S = \bigcup_{i=1}^{|S|} s_i$
$\forall g \in S : \sum_{i=1}^{|\Sigma|} E(g, z_i) = 1$, kde $\Sigma = \bigcup_{i=1}^{|\Sigma|} z_i$
$\sum_{i=1}^{|S|} A(s_i) = 1$, kde $S = \bigcup_{i=1}^{|S|} s_i$

Σ je množina znakov, ktoré daný HMM emituje v stavoch z S s pravdepodobnosťami E, P sú pravdepodobnosti prechodu zo stavu do stavu a A sú počiatočné pravdepodobnosti stavov.

Definícia 2

Skrytý Markovovský model vyššieho rádu je 7-ica $\{\Sigma, S, M, E, A, W, U\}$, kde W je zobrazenie $S \to N$ a E je zobrazenie $\Sigma^{W(q_i+1)} \times S \to R$, kde $q_i \in S$, a $U \in \Sigma^{max_{s \in S} W(s)}$ je počiatočná sekvencia ovplyvňujúca emisné pravdepodobnosti modelu, ostatné rovnako ako v predošlej definícii.
Platí:
$\forall q \in S, \forall k \in \Sigma^{W(q)} : \sum_{i=1}^{|\Sigma|} E(q, k + z_i) = 1$, kde $S = \bigcup_{i=1}^{|\Sigma|} z_i$

Teraz si definujeme zobrazenie, ktoré nám bude slúžiť ako premenná na uchovávanie ľubovoľnej distribúcie stavov počas výpočtu.

Definícia 3

Aktual A veľkosti n je usporiadaná n-tica reálnych čísel p_i predstavujúcich pravdepodobnosti 1. až n-tého stavu, pre ktorú platí:
$\Sigma_{i=1}^{n} p_i = 1$
Člena Aktualu A predstavujúceho pravdepodobnosť stavu q_i budeme označovať $A[q_i]$. Aktual skrytého Markovovského modelu M má veľkosť $|S|$, kde S je množina stavov modelu M.

Ešte nám ostáva zadefinovať si veci potrebné na simuláciu modelovaného procesu príslušným HMM. Udalosť, keď modelovaný proces emituje znak a následne prejde do nového stavu modelujeme krokom výpočtu HMM. Nová distribúcia stavov závisí od predošlej distribúcie stavov a emitovaného znaku. Krok výpočtu si definujeme ako zobrazenie s množiny kartézskeho súčinu množiny všetkých Aktualov a abecedy na množinu všetkých Aktualov.

Vykonaním sekvencie krokov na sekvencii DNA rozumieme iterovaný krok výpočtu HMM, kde ako prvý Aktual vezmeme počiatočnú distribúciu stavov a každý ďalší bude výsledkom zobrazenia predchádzajúceho Aktualu a príslušného znaku sekvencie DNA. I-tej iterácii prislúcha i-ty znak sekvencie DNA. Aktualom po vykonaní tejto sekvencie krokov rozumieme výsledok posledného takéhoto zobrazenia. Momentálnym Aktualom nazveme Aktual s distribúciou stavov po vykonaní príslušnej sekvencie krokov. Pravdepodobnosť emisie j-teho znaku závisí od momentálneho Aktualu daného HMM po vykonaní sekvencie krokov na predchádzajúcich j-1 znakoch danej sekvencie. Túto pravdepodobnosť definujeme ako zobrazenie s príslušnými vlastnosťami z množiny Aktualov. Teraz formálne:

Definícia 4
Pravdepodobnosť znaku z pri Aktuale A je zobrazenie $P : Q \times \Sigma \; toR$, kde Q je množina všetkých Aktualov, pre ktoré platí:
$P(A, z) = \sum_{s \in S} A[s] * E(s, z)$

Definícia 5
Krok výpočtu HMM je zobrazenie $K : Q \times \Sigma \to Q$, kde Q je množina všetkých Aktualov, pre K platí:
$\forall A \in Q, \forall z \in \Sigma : K(A, z) = B$, kde $\forall s \in S : B[s] = \sum_{q \in S} A[q] * M(q, s) * E(q, z)/P(A, z)$

Celková pravdepodobnosť sekvencie je vlastne pravdepodobnosť, že sa jej jednotlivé znaky objavia v správnom poradí na vstupe, teda je rovná súčinu pravdepodobností jej znakov v príslušných krokoch modelu na nej.

Definícia 6
Pravdepodobnosť sekvencie U pri distribúcii A je zobrazenie $P : A \times \Sigma^* \to R$, kde platí:
$P(U) = \prod_{i=1}^{|U|} P(A_i, U(i))$,
kde $A_1 = A$ a $A_i = K(A_{i-1}, z_{i-1})$ pre $\forall i \in N : 1 < i <= |U|$ a $U(i)$ je i-ty znak sekvencie U

2.2 HMM vyššieho rádu

Ako sme už uviedli, zaoberáme sa aj modelmi vyššieho rádu. Výhodou týchto modelov je, že nám umožňujú zadefinovať emisné pravdepodobnosti nielen vo vzťahu k stavu v ktorom sa model nachádza, ale aj vo vzťahu k už vypísanej časti sekvencie. Toto nám umožňuje zvýšiť presnosť výpočtu pravdepodobností jednotlivých znakov bez súčasného zvýšenia počtu stavov.

Emisné pravdepodobnosti v jednotlivých stavoch teda budú závisieť od predtým spracovanej sekvencie. Pre korektný začiatok výpočtu na sekvencii DNA teda potrebujeme nielen počiatočnú distribúciu stavov ale aj počiatočnú sekvenciu. Pokiaľ teda nevykonáme dostatočný počet krokov výpočtu na sekvencii DNA budeme suffix spracovanej sekvencie dopĺňať spredu suffixom počiatočnej sekvencie. Pri dostatočne dlhej sekvencii DNA voľba počiatočnej sekvencie celkovú pravdepodobnosť sekvencie veľmi

neovplyvní, vo väčšine prípadov si preto môžeme zvoliť sekvenciu, ktorá nám vyhovuje z hľadiska jednoduchosti výpočtu.

Teraz si zadefinujeme krok výpočtu a pravdepodobnosť znakov na vstupe pre HMM vyššieho rádu.

Definícia 7
Pravdepodobnosť emitovania znaku x pri predchádzajúcej sekvencii w a Aktualu A podľa HMM vyššieho rádu je zobrazenie $P : Q \times \Sigma^* \to R$, pre ktoré platí:
$P(A, wx) = \sum_{i=1}^{n} A[q_i] * E(q_i, u_i x)$;
kde u_i je posledných $W(q_i)$ znakov z w.

Definícia 8
Krok výpočtu HMM N, z Aktualu A, predošlej sekvencie w a emitovanom znaku z je zobrazenie $K : Q \times \Sigma^+ \to Q$, pre ktoré platí:
$K(A, wz) = B$, kde platí $\forall s \in S : B[s] = \sum_{q \in S} E(q, z) * A[q] * M(q, s)/P(A, wx)$

Pravdepodobnosť sekvencie vypočítame rovnakým spôsobom ako pri jednoduchých HMM. Keďže jednoduchý HMM je vlastne HMM nultého rádu, použitie HMM vyššieho rádu nemôže byť pri správnej voľbe vo všeobecnosti horšie. Potenciálnou nevýhodou HMM vyyššieho rádu sú len nároky na pamäť vyplývajúce z potreby uložiť väčšie množstvo pravdepodobností.

2.3 Trénovanie modelov

Medzi najťažšie a pritom najdôležitejšie pri práci s HMM patrí hľadanie a tvorba vhodných modelov. Tento proces má dve časti: prvou z nich je návrh štruktúry, teda koľko je stavov a ako sú pospájané, druhou je určenie pravdepodobností emisií a prechodov medzi stavmi. Budeme čerpať z [DEKM98].

Najľahšie je zostaviť model na dátach, v ktorých máme ku každej sekvencii všetky postupnosti stavov, ktoré ju môžu generovať. V tomto prípade len pre každý stav spočítame priemerné frekvencie emisitovania jednotlivých znakov a ako často z neho proces prešiel do ktorého stavu a z toho následne vypočítame priemerné pravdepodobnosti prechodov. Pri použití dostatočne reprezentatívnych dát vieme získať presný model daného procesu.

V prípade neznámej postupnosti stavov nie je presne určené priradenie parametrov

k jednotlivým stavom. Tu sa používajú iteračné metódy. Štandardne sa používa Baum-Welch algoritmus [Bau72]. Tento používa prirodzenú interpretáciu pravdepodobnosti. Iteruje sa do dosiahnutia určeného kritéria pričom sa dá dokázať konvergencia k lokálnemu maximu. Problémom je, že lokálnych maxím môže byť viacero a to do ktorého z nich sa dostaneme závisí od vstupných počiatočných nastavení. Toto je značná komplikácia pri konštrukcii veľa-stavových modelov.

Kapitola 3

Algoritmus a implementácia

V tejto kapitole sa budeme venovať použitým algoritmom, ich implementácii a špecifikám nášho programu. Najprv opíšeme algoritmus výpočtu pravedepodobností znakov na vstupe z HMM a algoritmus aritmetického kódovania, ktorý pomocou týchto pravdepodobností tieto znaky kóduje. Potom sa budeme venovať spojeniu týchto algoritmov do nášho programu a nakoniec uvedieme ostatné špecifiká nášo prístupu.

3.1 Algoritmus

Ako už bolo spomenuté, náš algoritmus pozostáva z dvoch hlavných častí: zo skrytého Markovovského modelu (HMM) ako modelu zdroja sekvencie, pomocou ktorého počítame pravdepodobnosti jednotlivých zankov a z aritmetického kódera (AC), ktorý ich na základe týchto pravdepodobností zakóduje. V prvej časti tejto podkapitoly sa budeme vanovať spôsobu výpočtu pravdepodobnosti sekvencie zo skrytého Markovovského modelu a následne ukážeme ako tieto pravdepodobnosti využijeme pri kompresii s použitím aritmetického kódovania. Skrytým Markovovským modelom modelujeme pravdepodobnosti, že budú nasledovať jednotlivé znaky. Pravdepodobnosť celej sekvencie teda vypočítame ako súčin pravdepodobností týchto znakov. Toto nám umožňuje kódovať sekvenciu na základe jej pravdepodobnosti postupne a využiť na to aritmetické kódovanie.

Používame klasický algoritmus aritmetického kódovania s adaptívnym modelom pravdepodobností znakov. Táto časť programu spracúva vstup. Podľa pravdepodobností znaku na vstupe ho zakóduje a následne zavolá časť s modelom o nové pravdepodobnosti znakov.

Pri dekompresii časť s AC spracúva sekvenciu bitov na vstupe a po každom rozkódovanom znaku zavolá model o pravdepodobnosti pre nasledujúci znak, až pokiaľ neprečíta celý vstup reprezentujúci komprimovanú sekvenciu genómu. Za predpokladu,

že je HMM pri dekompresii rovnaký a rovnako nainicializovaný ako pri kompresii, pracuje program korektne.

3.1.1 výpočet pravdepodobnosti z HMM - model

Pred samotným výpočtom je potrebné najprv načítať HMM a incializovať ho. Aby sme mohli korektne počítať a nemáme počiatočné rozdelenie stavov musíme si ho zvoliť. Ak počítame s modelom vyššieho rádu, tak si musíme zvoliť aj sekvenciu s ktoru budeme počítať ako s predchádzajúcou. V programe používame na reprezentáciu sekvencií čísla v pozičnej sústave, ktorej základom je veľkosť abecedy. Z tohto dôvodu sme si zvolili sekvenciu reprezentovanú číslom 0 ako predtým vypísanú sekvenciu. Pokiaľ sme nútení zvoliť si počiatočné rozdelenie stavov a nevieme nič o modely, zvolíme si rovnomernú distribúciu pravdepodobností, teda každému stavu priradíme rovnakú pravdepodobnosť. Týmto sa vyhneme komplikáciám, ktoré by nastali ak by graf možných prechodov medzi stavmi nebol orientovane súvislý.

Ako sme spomenuli vyššie, pravdepodobnosti s ktorými je HMM v jednotlivých stavoch sa po každom kroku menia, preto sme si zadefinovali premennú Aktual. V Aktuali A budeme uchovávať počiatočnú distribúciu stavov a v Aktuali B budeme uchovávať distribúúciu stavov v danom kroku výpočtu. Pri úvodnej inicializácii nastavíme hodnotu premennej B na hodnotu premennej A.

pseudokód inicializácie:

-načítaj tabuľky pravdepodobností prechodov a emisií
-načítaj rády stavov
-načítaj počiatočné rozdelenie stavov Aktual A
-inicializuj rozdelenie stavov na počiatočné rozdelenie Aktual $B := A$
-inicializuj premennú so suffixom vypísanej sekvencie sek $:= 0$

Keď je časť s HMM inicializovaná, môže byť zavolaná a korektne počítať pravdepodobnosti, že budú nasledovať jednotlivé znaky. Po načítaní znaku zo vstupu časť s AC pošle tento znak aj časti s HMM aby mohla vykonať podmienený krok výpočtu HMM. Keďže skrytý Markovovský model zodpovedá skrytému Markovovskému modelu vyššieho rádu s rádom nula, budeme vzorce uvádzať pre HMM vyššieho rádu. Pre jednoduchosť budeme o časti programu počítajúcej pravdepodobnosti znakov hovoriť ako o modely a o časti vykonávajúcej aritmetickékódovanie ako o kódery.

Ak je model zavolaný o pravdepodobnosti znakov $\{P(z_i)\}_{i=1}^{|\Sigma|}$, vypočíta ich nasle-

dovne:

$$P(z_j) = \Sigma_{i=1}^{|A|} B[s_i] * E(s_i, sekmod|\Sigma|^{W(s_j)} * |\Sigma| + z_j)$$

Po takomto volaní pošle kóderu vypočítané pravdepodobnosti, ten ich spracuje a pošle modelu informáciu o znaku ktorý zakódoval. Na základe tejto informácie model vykoná krok výpočtu a spočíta novú distribúciu stavov. Označme príslušný znak z. Výpočet kroku modelu pozostáva z dvoch častí: najprv model vypočíta podmienenú pravdepodobnosť stavov $\{C[s_i]\}_{i=1}^{|P|}$ nasledovne:

$C[s_i] = B[s_i] * E(s_i, sekmod|\Sigma|^{W(s_j)} * |\Sigma| + z_j)/P(z)$; kde P(z) je pravdepodobnosť znaku z počítaná podľa predošlého vzorca.

Model vypočíta novú distribúciu stavov z pravdepodobností prechodov medzi stavmi a práve vypočítanej podmienenej distribúcie stavov. Novú distribúciu D teda vypočítame ako:

$$D[s_i] = \Sigma_{j=1}^{|C|} C[s_j] * M(s_j, s_i)$$

Túto distribúciu následne priradíme do Aktualu B a týmto sme skončili výpočet kroku modelu. Keď bude model najbližšie zavolaný, aby dodal pravdepodobnosti znakov, budú to už pravdepodobnosti nasledujúceho znaku.

Pre prípad komprimovania viacerých nesúvisiacich sekvencií rovnakým modelom je implementovaná funkcia reset, ktorá inicializuje model do počiatočného stavu. Túto funkciu sme využili aj pri testovaní jednotlivých modelov, keď program skomrimovanú sekvenciu genómu vzápätí dekomprimoval kvôli kontrole správnosti.

3.1.2 Aritmetické kódovanie - kóder

V tejto časti opíšeme použitý algoritmus aritmetického kódovania. Myšlienku aritmetického kódovania sme už spomenuli v časti o kompresii. Ako sme uviedli výhody aritmetického kódovania sú, že nám umožňuje kódovať sekvenciu celú – nemusíme ju deliť do blokov, ďalej to, že nemusíme kódovať znaky celým počtom bitov ale môžem napríklad bázu A zakódovať 0,47 bitmi. Medzi výhody aritmetického kódovania patrí taktiež jeho rýchlosť, ktorá je spôsobená jeho jednoduchosťou a tým, že využíva len jednoduché aritmetické operácie.

Najprv sa budeme venovať kompresii. Ako bolo uvedené, z pohľadu aritmetického kódovania prislúcha každej sekvencii interval, ktorého dĺžka je priamo úmerná pravdepodobnosti, s ktorou sa táto sekvencia vyskytne na vstupe. Referenčným intervalom budeme nazývať interval, ktorý prislúcha doteraz prečítanému (pri kompresii) alebo doteraz dekódovanému (pri dekompresii) prefixu kódovanej sekvencie. Kódovanou sekvenciou budememe volať sekvenciu, ktorú komprimujeme respektíve, ktorrej zápis dekomprimujeme.

kompresia:
Najprv nastaví referenčný interval na počiatočnú veľkosť, teda <0,1), a model zdroja inicializuje zavolaním príslušnej funkcie do počiatočného stavu. Zatiaľ sme zapísali ϵ a tomu zodpovedá aj referenčný interval. Všetko je teda pripravené na kódovanie.

inicializácia – pseudokód:

horná hranica refrenčného intervalu $hh = 1,0$
dolná hranica refrenčného intervalu $dh = 0,0$
inicializuj model

Nasleduje cyklus v ktorom bude kóder postupne čítať a kódovať znaky zo vstupu. Teda pozná čoraz dlhší prefix kódovanej sekvencie a podľa neho čoraz presnejšie špecifikuje referenčný interval, teda interval v ktorom leží interval prislúchajúci tejto sekvencii.

Znakom, ktoré sa môžu objaviť na vstupe, sú priradené navzájom disjunktné podintervaly referenčného intervalu, ktoré tento interval zároveň pokrývajú. Následne prečíta znak zo vstupu a jeho podinterval vezme za nový referenčný interval. Keďže tieto podintervaly referenčného intervalu sú jednotlivým znakom priraďované na základe ich pravdepodobnosti a pravdepodobnosť sekvencie je rovná súčinu pravdepodobností s ktorými sa na vstupe postupne objavia jej jednotlivé znaky, bude veľkosť referenčného intervalu po prečítaní celej sekvencie priamo úmerná jej pravdepodobnosti.

Ako sme už uviedli cieľom kódera je zakódovať interval čo najkratším číslom z neho. Preto keď sa nám interval dostatočne zmenší a jeho horná aj dolná hranica majú prvé bity svojich zápisov rovnaké, môžeme tieto bity dať na výstup a zabudnúť ich, pretože všetky čísla z tohto intervalu ich musia mať rovnaké. Toto nám umožňuje kódovať interval, ktorý pripadne kódovanej sekvencii priebežne spolu s jej čítaním, pričom pamäť je vzhľadom vstupnú sekvenciu konštantná.

Niekedy sa môže stať, že dostaneme dolnú hranicu tvaru 0111www a hornú tvaru

1000zzz. V tomto prípade nevieme aký dať výstup a strácame aj presnosť hraníc referenčného intervalu. Riešením je odložiť jednotky z dolnej hranice a nuly z hornej a zapamätať si počet takto odložených bitov. Teda dolná hranica bude 0www a horná 1zzz. Keď sa najbližšie zhodnú hranice intervalu, budeme vedieť koľko a aké bity dať na výstup [STANEK].

cyklus pre každý znak kódovanej sekvencie – pseudokód

B

Zistí z modelu dátového zdroja pravdepodobnosť s ktorou sa jednotlivé znaky objavia na vstupe a rozdelí podľa nich referenčný interval na intervaly sekvencií o jedna dlhších ako doteraz prečítaná sekvencia, ktorým je táto sekvencia prefixom.

zavolaj model o pravdepodobnosti znakov $\{f_i\}$
rozsah referenčného intervalu $rr = hh - dh$
$h_0 = dh$
pre $i = 1$ do veľkosť abecedy

B

$h_i = h_{i-1} + f_i * rr$

E

Prečíta zo vstupu znak a zoberie za nový referenčný interval interaval prislúchajúci zreťazeniu doteraz prečítaného prefixu a tohto znaku.
zo vstupu načítaj znak Z
priraď mu číslo z abecedy $C = abeceda[Z]$
uprav referenčný interval na podinterval tohto znaku
$dh = h_{C-1}$
$hh = h_C$
Postupne ako sa referenčný interval zmenšuje, sú vrchné bity jeho hornej a dolnej rovnaké a teda vieme, že každé číslo z referenčného intervalu ich bude mať takéto. Preto ich priebežne vypisujeme a v hornej a dolnej hranici referenčného intervalu si uchovávame len suffix hornej a dolnej hranice intervalu doteraz prečítaného prefixu kódovanej sekvencie.

kým hh-dh<0,5 rob

B
ak $dh > 0,5$ vypíš prenesené bity (pb) a 1
inak ak $hh < 0,5$ vypíš prenesené bity a 0
inak zvýš počet prenesených bitov $inc(pb)$
$hh = hh * 2$
$dh = dh * 2$
E

E

terminácia
Na konci výpočtu potrebujeme dopísať suffix čísla z intervalu prečítanej sekvencie, aby sme ho vedeli jednoznačne dekódovať pri dekompresii. Aby sme daný interval vedeli pri dekompresii jednoznačne určiť zapíšemem dolnú hranicu najväčšieho intervalu, ktorého hranicami sú celé záporné mocniny dvojky a leží celý v referenčnom intervale. Keďže z programu vyplýva, že $rr > 0,5$*, bude to buď* $0,00; 0,25 alebo 0,50$
ak $dh = 0$ vypíš $0,00$
inak ak $dh < 0,25$ vypíš $0,25$
inak vypíš $0,50$

dekompresia:
Pri dekompresii má program zakódovaný interval ako reťazec bitov predstavujúci číslo z neho a chce zistiť, ktorej sekvencii patrí. Bude mať preto pomocný interval, ktorý bude najväčší taký interval, ktorý má prefixy hraníc rovnaké ako doteraz prečítaný prefix reťazca bitov. Aby šetril miesto, bude si rovnako ako aj pri referenčnom intervale aj v tomto prípade uchovávať len potrebné suffixy oboch hraníc intervalu. Akonáhle padne pomocný interval do podintervalu niektorého zanku, môže program tento znak vypísať, lebo určite bude ďalším znakom prefixu sekvencie, ktorej načítavaný interval prislúcha. V tomto prípade upraví referenčný interval na interval daného znaku, zahodí nepotrebný prefix pri hraniciach referenčného a pomocného intervalu; zistí z modelu nové pravdepodobnosti znakov a rozdelí referenčný interval na podintervaly pre jednotlivé znaky. Teda postupuje podobne ako pri kompresii až na to, že tentoraz

zo zapísaného čísla zisťuje referenčný interval do ktorého toto číslo patrí a podľa neho skomprimovanú sekvenciu.

inicializácia

horná hranica refrenčného intervalu $hh = 1,0$
horná hranica pomocného intervalu $ph = 1,0$
dolná hranica refrenčného intervalu $dh = 0,0$
dolná hranica pomocného intervalu $pd = 0,0$
inicializuj model
zavolaj model o fekvencie znakov $\{f_i\}$
rozsah referenčného intervalu $rr = hh - dh$
$h_0 = dh$
pre $i = 1$ do veľkosť abecedy
B
$h_i = h_{i-1} + f_i * rr$
E

kým neskončí vstupný súbor rob:
B
Zmenší pomocný interval podľa bitu na vstupe
rozsah pomocného intervalu $pr = hh - dh/2$
ak je na vstupe 0 $ph = ph - pr$
inak $pd = pd + pr$
skontroluj, či pomocný interval nepadol celý do intervalu niektrého znaku
ak áno (označme číslo daného znaku C)
B
úprava referenčného intervalu
$dh = h_{C-1}$
$hh = h_C$
zahodenie nepotrebných prefixov hraníc intervalov
kým $hh - dh < 0,5$ rob
B
$hh = hh * 2$
$dh = dh * 2$
$ph = ph * 2$
$pd = pd * 2$

E
rozdelenie referenčného intervalu na podintervaly pre jednotlivé znaky
zavolaj model o fekvencie znakov $\{f_i\}$
rozsah referenčného intervalu $rr = hh - dh$
$h_0 = dh$
pre $i = 1$ do veľkosť abecedy
B
$h_i = h_{i-1} + f_i * rr$
E
E
E
terminácia:
Dekompresia skončí keď program prečíta celú sekvenciu bitov reprezentujúcu komprimovanú sekvenciu.

Pri kompresii horeuvedeným spôsobom bude posledný dekomprimovaný znak zároveň aj posledným znakom skomprimovanej sekvencie. Teda dekomprimovaná sekvencia bude zodpovedať skomprimovanej. Toto je zaistené tým, že model dá pre rovnakú distribúciu stavov rovnaké pravdepodobnosti pre znaky na vstupe a pre rovnakú distribúciu a rovnaký znak vykoná podmienený krok výpočtu s rovnakým výsledkom. Pre danú sekvenciu pracuje teda model deterministicky a môžeme sa spoľahnúť, že pri dekompresii bude počítať rovnaké pravdepodobnosti ako pri kompresii a rovnakej sekvencii priradí rovnaký podinterval intervalu $<0,1)$.

3.2 Implementácia

Program je písaný v jazyku $C++$. Stavy modelu a emitované znaky reprezentujeme celými číslami, čo nám uľahčuje indexovanie. Pri znakoch sa pridáva výhoda ľahšieho získavania prefixov a suffixov. Keďže sekvencie sú vlastne čísla v sústave, ktorej základom je veľkosť abecedy, vieme podreťazec ľubovoľnej sekvencie získať vhodným celočíselným delením a moduláciou sekvencie.

3.2.1 Implementácia HMM

K HMM má program uložené tabuľky pravdepodobností ako matice v poliach príslušného rozmeru. Keďže veľkosť modelu sa môže meniť uchováva si ich v podobe smerníkov ku ktorým si alokuje pamäť potrebnej veľkosti. Pre uloženie pravdepodobností používa program nami definovanú triedu Číslo, ktorej sa venujeme nižšie.

Pre výpočet pravdepodobností znakov z aktuálnej distribúcie stavov, počítanie podmieneného kroku výpočtu HMM a inicializáciu sú definované vlastné funkcie a procedúry na modely. Taktiež je definovaná funkcia reset na inicializáciu modelu do počiatočného stavu. Načítanie modelu je robené cez procedúru triedy Model, ktorá dostane pointer na príslušný stream, kde sa predpokladá, že na ňom nasledujú údaje o modely v dohodnutom poradí.

3.2.2 Implementácia aritmetického kódera

Aritmetický kóder je upravený a jeho pôvodný zdroják pochádza zo stránky [Whe]. Z tejto stránky sme prevzali zdroják aritmetického kódovania s jednoduchým adaptívnym modelom frekvencií. Teda modelom, ktorý ich upravuje na základe frekvencie v už prečítanom reťazci. Tento model sme odstránili a kód sme upravili tak, aby pravdepodobnosti písmen boli počítané pomocou HMM. V použitom riešení sú hranice intervalov uchovávané ako celé čísla veľkosti longint. Toto umožnilo v pôvodnej implemntácii vyhnúť sa reálnej aritmetike.

Naša úprava spočívala v prispôsobení funkcií na volanie adaptívneho dátového modelu, konkrétne procedúry inicializácie a procedúr na získanie momentálnych pravdepodobností znakov na vstupe a úpravy modelu na základe práve prečítaného znaku. Hranice podintervalov sa prideľujú podľa vypočítaných frekvencií vzhľadom na interval $<0,1)$ a následne sa škálujú podľa veľkosti referenčného intervalu.

3.3 Špeciálna numerika

Náš program vykonáva aritmetické operácie s malými číslami a tiež sčituje väčšie množstvo malých čísel, implementovali sme špecifickú aritmetiku. Pravdepodobnosti a reálne čísla vôbec máme v logaritmickej škále, to znamená, že namiesto čísla pracujeme s jeho logaritmom pri prirodzenom základe. Toto nám umožnilo pohodlnejšie počítať súčin a podiel a taktiež sčitovať bez straty presnosti malé čísla.Využili sme tieto vzorce:

$$log(a * b) = log(a) + log(b)$$

$$log(a/b) = log(a) - log(b)$$

$$log(a + b) = log(1 + e^{log(a/b)}) + log(b) \text{ ; kde b>=a}$$

Za týmto účelom sme implementovali vlastnú triedu Číslo a definovali sme pre ňu príslušné operátory. Taktiež sme boli pre túto triedu nútení implementovať porovná-

vacie operátory, konštruktory a deštruktory. Následne sme pre správnosť výpočtu definovali aj niektoré metódy štruktúry map pre prácu s prvkami tejto triedy.

V našom algoritme sčítavame veľké množstvo čísel, ktoré sú voči konečnému výsledku relatívne malé a ich púhe pričitovanie k výsledku by nás stálo stratu presnosti. Preto sme implementovali funkciu na sčítavanie veľkého počtu čísel v logaritmickej škále, ktoré sme predtým vložili do haldy s najmenším číslom navrchu. Táto funkcia sčítava vždy dve najmenšie čísla, ktoré postupne vyberá z vrchu haldy a výsledok vloží späť. Tento postup opakuje dovtedy, kým nie je halda prázdna.

Zvolený prístup k sčítavaniu zabezpečuje, že výsledok bude obsahovať len chybu zo zaokrúhlenia, ktorá je vzhľadom na použitie logaritmickej reprezentácie čísla a presnosť modelov zanedbateľná.

Kapitola 4

HMM ako model genetickej sekvencie

V tejto kapitole sa budeme venovať možnosti modelovať genóm pomocou skrytých Markovovských modelov (HMM). Na základe biologického výskumu vieme, že genóm nie je homogénny ale skladá sa z úsekov s rôznych typov s navzájom odlišnými vlastnosťami a funkcionalitou. Od typu úseku genómu závisí štruktúra DNA, teda pravdepodobnosti a pravidelnosti v poradí a rozmiestnení jednotlivých báz. V tomto prípade sa na sekvenciu genómu pozrieme ako na tok dát produkovaný procesom, ktorý môže byť v rôznych stavoch. Tento proces budeme následne modelovať skrytým Markovovským modelom. Jednotlivé homogénne úseku genómu budeme modelovať stavmi. Tento model následne natrénujeme na reálnych dátach. Ako sme už spomenuli, čím viac toho o štruktúre a charaktere dát vieme, tým lepší HMM vieme navrhnúť a natrénovať. Zo získaného kompresného pomeru sa potom môžeme dozvedieť ako dobre sú vedomosti, na základe ktorých sme vytvorili príslušný HMM, vystihujú sekvenciu genómu.

4.1 Modelované vlastnosti

4.1.1 Značkované sekvencie

Asi naznámejšou heterogenitou v DNA je jej zloženie z úsekov nesúcich informáciu o niektorej dedičnej vlastnosti, takzvaných génov, a z úsekov, ktoré túto informáciu nenesú, zvaných medzigénové (intergenic). Gén sa môže skladať z viacerých kódujúcich úsekov. To sú úseky nesúce informáciu o skladbe bielkovinových sekvencií. Táto informácia je uložená v poradí a druhu jednotlivých trojíc báz – tripletov. Pričom každý triplet hovorí o umiestnení jednej aminokyseliny.

Proces syntézy bielkoviny, nazývaný translácia, spočíva z niekoľkých fáz. Najprv sa informácia nesená molekulou DNA v jej kódujúcej časti prepíše do molekuly kyseliny ribonukleovej (RNA). Z nej sa následne niektoré časti, zvané intróny,vystrihnú

a odstránia. Častiam, ktoré po tomto procese ostanú hovoríme exóny. Ku jednotlivým tripletom v takto vzniknutej sekvencii sa následne pripojí komplementárna molekula RNA naviazaná na príslušnú aminokyselinu. Nakoniec príslušný enzým spojí jednotlivé molekuly aminokyselín do bielkovinového reťazca.

Vďaka výskumu tohto procesu existujú označkované sekvencie DNA, teda sekvencie o ktorých vieme, ktoré úseky sú kódujúce, ktoré nekódujúce a o kódujúcich aj či ide o exóny alebo intróny a taktiež pozíciu bázy v triplete.

Na základe týchto vedomostí sme vytvoriliviacero modelov. Najjednoduchší model modeloval len to či ide o úsek sekvencie genómu s génom alebo nie. Najzložitejší obsahoval stav pre každú značku, teda spolu 17 stavov.

Vytvorili sme program na trénovanie modelov z označkovaných sekvencií a jednotlivé modely. Po vytvorení týchto modelov sme ich trénovali na označkovaných sekvenciách. Ak je sekvencia označkovaná, existuje k nej rovnako dlhá sekvencia značiek, kde i-temu znaku sekvencie prislúcha i-ty label. Pre každý model sme najprv zadefinovali, akému stavu prislúcha ktorý label.

Následne sme pre každý model spustili trénovací program. Tento spočítal pre každý stav frekvenciu jednotlivých písmen a taktiež ako často za ním nasledoval ktorý stav a z toho vypočítal pre každý stav frekvenciu prechodu do toho–ktorého stavu. Tieto frekvencie sme následne vzali ako pravdepodobnosti pre náš model. Trénovali sme modely nultého rádu. Ako počiatočné rozdelenie sme zvolili, že začíname v časti sekvencie nepatriecej žiadnemu génu.

Takto natrénovaný model sme potom použili na kompresiu inej sekvencie DNA. Dáta na ktorých sme trénovali a následne testovali náš model pochadzaju z organizmu Drosophila melanogaster (vínna muška). Konkrétne ide o chromozómy 2L a 2R, pričom boli rozdelené na trénovaciu a testovaciu množinu.Označenie báz sme zobrali z databazy RefSeq. Pričom sme odstránili prekrývajúce sa značky.

Naše vstupné dáta obsahovali tieto značky: x - časť sekvencie neobsahujúca gén d,a,i - intróny na doprednom vlákne b,0,1,2,e - exóny na doprednom vlákne D,A,I - intróny na spätnom vlákne B,3,4,5,E - exóny na spätnom vlákne

Kde čísla označujú pozíciu bázy v triplete: 0 a 3 je prvá báza, 1 a 4 je druhá a 2 a 5 je tretia báza. Každý intrón je ohraničený bázami, ktoré označujú miesto, kde má byť sekvencia prestrihnutá, tieto bázy nazývame signál. V našom labelovaní sú signálmi d,i,b,e,D,I,B a E. Pri translácii DNA sú možné dva smery: dopredný a spätný, čo zachytáva aj naše rozlíšenie labelov medzi dopredným a spätným vláknom.

Ako sme už spomenuli, náš prvý model pozostával z dvoch stavov: kódujúceho (značka x) a nekódujúceho (ostatné značky, okrem x). Pmocou tohto natrénovaného modelu sme dosiahli na testovacej sekvencii kompresný pomer 1,9717 bitu na bázu.

Postupne sme zvyšovali počet stavov v modely, pričom sme značky s podobným význam zružovali do jedného stavu. So zvyšovaním počtu stavov v trénovanom modely sa zároveň zlepšoval dosiahnutý kompresný pomer.

Ako posledný sme trénovali model pri ktorom sme vytvorili pre každú značku iný stav. S týmto modelom sme po natrénovaní a testovaní na príslušných sekvenciách dosiahli kompresný pomer 1,9676 bitu na bázu.

4.1.2 Frekvencia báz C a G

Ďalšou vlastnosťou je výskyt úsekov s vysokou frekvenciou báz C a G a úsekov kde je naopak veľmi nízka. Najprv sme túto vlastnosť modelovali modelom poostávajúcim z troch stavov. Prvý stav modelujoval úseky s nízkym výskytom báz C a G, druhý modelujoval úseky s priemernou frekvenciou týchto báz a tretí modelujoval úseky s vysokou frekvenciou týchto báz. Trénovací program pre tento model vypočítal pomer výskytu báz C a G voči počtu báz v okne zvolenej dĺžky a na základe tohto pomeru priradil stav pozícii v strede okna. Ako sme už spomenuli okno je súvislý podreťazec sekvencie. S takýmto programom sme následne natrénovali viacero modelov, pričom sme menili veľkosť okna a veľkosť pomerov pre jednotlivé stavy.

Najprv sme zvolili pre prvý stav hornú hranicu 0,5 a pre druhý stav 0,7. Následne sme menili veľkosť okna. Pri okne veľkosti 7 báz sme dosiahli kompresný pomer 1,9825 bitov na bázu. Po zväčšení okna na 15 báz sa pomer zlepšil na 1,9780 bitov na bázu a po zväčšení okna na 31 sa kompresný pomer zmenšil na 1,9759 bitov na bázu.

Z tohto sme zistili, že na kompresiu pomocou skrytých Markovovských modelov sú v tomto prípade využiteľnejšie stavy reprezentujúce zmeny vo frekvencii báz C a G na dlhšom úseku genómu, ako tie, ktoré modelujú lokálne zvýšenie frekvencie týchto báz na krátkom úseku.

Taktiež sme si všimli, že pri zmene hraníc frekvencie báz C a G v jednotlivých stavoch sa menila početnosť týchto stavov na trénovanej sekvencii. Aby sme mohli tento jav lepšie skúmať, vytvorili sme model obsahujúci 19 stavov. Horná hranica prvého stavu bola 0,1 a každého ďalšieho o 0,5 vyššia ako pri predchádzajúcom stave. S týmto modelom sme dosiahli kompresný pomer 1,9743 bitu na bázu.

Zaujímavejšie však bolo v tomto prípade zistenie o početnosti stavov. Pri oknách menšej veľkosti bol zastúpený prvý stav, niekoľko nasledujúcich stavov nebolo zastúpených vôbec a prvým zastúpeným bol stav, ktorý reprezentoval frekvenciu báz C a G od 0,6. Pri postupnom zvyšovaní veľkosti okna sa početnosť prvej polovice stavov na začiatku nemenila ale už pri veľkosti okna 95 neboli zastúpené posledné dva stavy. Z tohto možno usudzovať, že trénovacia sekvencia DNA obsahovala dlhšie úseky takmer výhradne z báz A a T a zvyšné úseky mali vyššiu frekvenciu báz C a G pričom

obsahovali krátke úseky zložené takmer výlučne z báz C a G. Tieto úseky však boli kratšie ako podobné úseky báz A a T.

Následne sme sa ešte pokúšali zlepšiť kompresný pomer trénovaním HMM vyšších rádov. Pri trénovaní stavov podľa frekvencie báz Ca G sme zaznamenali sotva badateľné zlepšenie, avšak pri označkovaných sekvenciách sme zaznamenali zhoršenie kompresného pomeru až o bit na bázu. Rovnako nám nepomohlo ani zvýšenie počtu stavov. Z toho usudzujeme, že tvorba HMM na kompresiu genómu si vyžaduje komplexnejší prístup.

4.2 Diskusia o výsledkoch a cieľoch

Hlavnou prekážkou v našej práci bola neexistencia vhodných HMM pre sekvenciu DNA. Modely trénované na označkovaných sekvenciách primárne používané na vyhľadávanie génov sa ukázali ako nie veľmi vhodné na kompresiu genetickej sekvencie.

Podobne dopadla aj snaha o kompresiu na základe frekvencie výskytu báz C a G v okolí, využívajúca výskyt takýchto úsekov v DNA. V tomto prípade sa ako pomocou HMM vystihnuteľnejšia ukázala zmena tejto frekvencie v širšom okolí.

Na druhej strane náš program môže poslúžiť na testovanie toho, ako dobre takéto modely dokážu predpovedať pravdepodobnosť konkrétnych báz na jednotlivých pozíciách v genetickej sekvencii.

Pre budúcnosť by sme chceli poukázať na možnosť pomôcť si pri tvorbe HMM ich ďalšou špecifíkaciou – dynamickými HMM. Dynamické HMM sú modely, ktoré sa čoraz častejšie používajú pri entropickej kompresii ťažko komprimovateľných dát. Ich výhodou je, že sa dynamicky prispôsobujú sekvencii počas jej načítavania. Nevýhodami sú veľká spotreba pamäte a pomalý beh algoritmu.

Domnievame sa, že by sa takýto dynamický HMM vytvorení na sekvencii DNA dal využiť pri tvorbe HMM, ktorý by bol jednoduchší a statický, čím by sme odstránili hlavné nevýhody dynamického HMM.

Záver

V našej práci sme implementovali nový prístup ku kompresii sekvencie DNA pomocou aritmetického kódovania s adaptívnym modelom. Ako typ modelu sme vybrali skrytý Markovovský model (HMM), pretože jednoduché pravdepodobnstné modely nedokázali dostatočne vystihnúť sekvenciu DNA. Špecifikom tejto sekvencie totiž je, že celková frekvencia báz je veľmi podobná.

Avšak molekula DNA nie je homogénna a skladá sa z mnohých typov úsekov s navzájom rôznymi frekvenciami jednotlivých báz. Tieto úseky sme sa snažili čo najpresnejšie modelovať stavmi HMM, pričom sme si všímali ich rôzne vlastnosti a podľa nich sme ich priraďovali jednotlivým stavom.

Naše výsledky zatiaľ nedosiahli úroveň kompresného pomeru algoritmov špecializovaných na DNA, ale veríme, že existuje potenciál zlepšiť ho použitím stále vhodnejších skrytých Markovovských modelov.

Pre náš prístup sme navrhli a implementovali program, ktorému sme pridali rôzne nuansy za účelom zlepšenia jeho práce s malými číslami.

Pôvodne sme sa pokúšali implementovať aj algoritmus Huffmanovho kódovania. Jeho implementáciu sme však nedokončili, pretože sa ukázal ako nevhodný pre svoju časovú náročnosť, zložitosť a potrebu komprimovať sekvenciu v blokoch, čo nás, ako sme spomenuli v kapitole o kompresii, núti požiť na každý blok sekvencie celočíselný počet bitov. Napriek možnosti výsledok čiastočne predpočítať sa táto cesta ukázala ako neperspektívna a preto sme sa jej v našej práci nevenovali, aj keď nám zabrala najviac času.

Literatúra

[Bau72] L. E. Baum. An equality and associated maximization technique in statistical estimation for probabilistic functions of markov processes. *Inequalities*, 3:1–8, 1972.

[BFG] Adam L. Buchsbaum, Glenn S. Fowler, and Raffaele Giancarlo. Improving table compression with combinatorial optimization. *J. ACM*, 50(6).

[BLF] Behshad Behzadi and Fabrice Le Fessant. DNA Compression Challenge Revisited: A Dynamic Programming Approach. In Alberto Apostolico, Maxime Crochemore, and Kunsoo Park, editors, *Combinatorial Pattern Matching*, volume 3537 of *Lecture Notes in Computer Science*, pages 85–96. Springer Berlin / Heidelberg.

[BWBW94] M. Burrows, D. J. Wheeler, M. Burrows, and D. J. Wheeler. A block-sorting lossless data compression algorithm. Technical report, 1994.

[CDAM07] Minh Duc Cao, T.I. Dix, L. Allison, and C. Mears. A Simple Statistical Algorithm for Biological Sequence Compression. In *Data Compression Conference, 2007. DCC '07*, 2007.

[CH87] G. V. Cormack and R. N. S. Horspool. Data Compression Using Dynamic Markov Modelling. *The Computer Journal*, 30(6):541–550, 1987.

[CKL00] Xin Chen, Sam Kwong, and Ming Li. A compression algorithm for DNA sequences and its applications in genome comparison. In *Proceedings of the fourth annual international conference on Computational molecular biology*, RECOMB '00, pages 107–, New York, NY, USA, 2000. ACM.

[CL04] Neva Cherniavsky and Richard Ladner. Grammar-based Compression of DNA Sequences, 2004.

[CW84] J. Cleary and I. Witten. Data Compression Using Adaptive Coding and Partial String Matching. *Communications, IEEE Transactions on*, 1984.

[DEKM98] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchinson. *Biological sequence analysis, Probabilistic models of protein and nucleic acids.* Cambridge University Press, 1998.

[Deo03] S. Deorowicz. Universal lossless data compression algorithms, 2003.

[GSU09] Raffaele Giancarlo, Davide Scaturro, and Filippo Utro. Textual data compression in computational biology: a synopsis. 2009.

[GT93] S. Grumbach and F. Tahi. Compression of DNA sequences, 1993.

[GT94] Stéphane Grumbach and Fariza Tahi. Compression of DNA sequences (Extended Abstract), 1994.

[HH94] K. Holtz and E. Holtz. Lossless data compression techniques. In *WESCON/94. 'Idea/Microelectronics'. Conference Record*, 1994.

[KPZ] Shanika Kuruppu, Simon Puglisi, and Justin Zobel. Relative Lempel-Ziv Compression of Genomes for Large-Scale Storage and Retrieval.

[MNSV] Veli Mäkinen, Gonzalo Navarro, Jouni Sirén, and Niko Välimäki. Storage and Retrieval of Individual Genomes.

[Say96] Khalid Sayood. *Introduction to Data Compression.* Academic Press, 1996.

[Sha48] Claude Shannon. A mathematical theory of communication. *The Bell Systems Technical Journal*, 1948.

[Whe] Fred Wheeler. http://www.fredwheeler.org/ac/.

[WST95] F.M.J. Willems, Y.M. Shtarkov, and T.J. Tjalkens. The context-tree weighting method: basic properties. *Information Theory, IEEE Transactions on*, 41(3):653 –664, may 1995.

[ZL77] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *Information Theory, IEEE Transactions on*, 1977.

[ZL78] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *Information Theory, IEEE Transactions on*, 1978.