Kompression von DNA Sequenzen

Michael Hiller

Diplomarbeit

Fakultät für Mathematik und Informatik
der Universität Leipzig

Studiengang medizinische Informatik

Leipzig, 5. Juni 2002

Kurzzusammenfassung

Standardkompressionsverfahren erreichen bei Texten, die Programmkode oder menschliche Sprache enthalten, sehr hohe Kompressionsraten. DNA Sequenzen weisen eine lineare Basenabfolge auf und können daher auch als Texte betrachtet werden. Allerdings gelingt es den Standardkompressoren in den meisten Fällen nicht, DNA Sequenzen zu komprimieren. Selbst wenn sich eine DNA Sequenz komprimieren läßt, ist die Kompressionsrate äußerst gering.

Es gibt zwei mögliche Erklärungen für diese Beobachtung. Entweder sind DNA Sequenzen generell nicht komprimierbar oder die bisherigen Kompressoren sind dazu nicht in der Lage. In dieser Diplomarbeit soll diese Frage untersucht werden. Dabei wird sich herausstellen, daß sich auch DNA Sequenzen komprimieren lassen, wenn ein Kompressor charakteristische Eigenschaften ausnutzt. Es werden wesentliche Unterschiede von menschlicher Sprache zu DNA Sequenzen erläutert und DNA spezifische Kompressionsverfahren vorgestellt.

Kompression läßt sich nicht nur zur Reduktion des benötigten Speichers verwenden, sondern es gibt eine Vielzahl an Anwendungsgebieten, wo Kompressionsverfahren in der Biologie eingesetzt werden können. So lassen sich phylogenetische Bäume mit Hilfe von Kompressionsverfahren aus genetischen Daten rekonstruieren. Auch zur Charakterisierung von unterschiedlichen Regionen im Genom kann ein Kompressor verwendet werden. Im zweiten Teil der Arbeit wird auf diese Gebiete näher eingegangen. Dabei werden praktische Versuche durchgeführt, um Nutzen und Anwendbarkeit genauer zu untersuchen, aber auch um Grenzen der Methoden aufzuzeigen.

Danksagung

Ich möchte allen an dieser Diplomarbeit Beteiligten herzlich danken. Besonderer Dank gebührt Herrn Dr. Raouf Hamzaoui und Herrn Dr. Knut Krohn für die gute Betreuung, hilfreiche Ratschläge und die Bereitschaft, sich mit all meinen Fragen und Problemen auseinanderzusetzen. Herrn Prof. Dietmar Saupe danke ich für diese sehr interessante Aufgabenstellung.

Desweiteren möchte ich Herrn Prof. Arndt von Haeseler für die nette Zusammenarbeit, viele fruchtbare Diskussionen sowie für wertvolle Hinweise und Anregungen ganz herzlich danken.

Sehr herzlich danke ich auch meiner Familie, die mir mein Studium ermöglicht und mich stets nach Kräften unterstützt hat.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einführung

Datenkompression spielt eine wichtige Rolle im heutigen Informationszeitalter. Um Daten effizient übertragen oder speichern zu können, versucht man die Redundanzen einer Datei zu entfernen, um dadurch die ursprüngliche Größe einer Datei zu verkleinern. Man unterscheidet zwischen verlustbehafteter Kompression, die vor allem auf Bild–, Video– und Audiodaten angewendet wird und keine originalgetreue Rekonstruktion erlaubt sowie der verlustfreien Kompression, die man bei Texten, Programmdateien und anderen Daten, bei denen eine exakte Rekonstruktion nötig ist, verwendet. In dieser Diplomarbeit wird sich nur mit verlustfreier Kompression beschäfigt.

Für die Kompression von Texten, die menschliche Sprache enthalten, wurde eine Reihe von hocheffizienten Methoden entwickelt, die heute zur Standardsoftware eines jeden Rechners gehören und ein breites Anwendungsspektrum haben. So sind die meisten Dateien, die aus dem Internet heruntergeladen werden können, mit *gzip* (Endung .gz) oder vergleichbaren Programmen gepackt. Diese Dateien beeinhalten typischerweise Texte oder Programme (als Quell– oder Maschinenkode) und damit eine Menge Redundanz. Ein Kompressor entfernt diese Redundanz, wodurch die Dateigröße sinkt.

Wie effizient solche Verfahren sind, soll kurz demonstriert werden. Als Beispiel für menschliche Sprache dient der gesamte Text dieser Diplomarbeit, der 281463 Bytes benötigt. In dieser Datei befinden sich aber nur 94 verschiedene Zeichen (alle druckbaren ASCII Zeichen). Daher benötigt man nur $\lceil log_2 94 \rceil = 7$ Bits für ein Zeichen. Die korrigierte Dateigröße ist damit $281463 * 7/8 = 246281$ Bytes. Nach der Kompression mit *gzip* benötigt die Datei nur noch 85122 Bytes, was einer Kompressionsrate von 65 % entspricht.

Eine DNA Sequenz kann aufgrund der linearen Basenabfolge ebenfalls als

Text betrachtet werden. Als Beispiel für eine DNA Sequenz wird ein 219447 Basen langer Abschnitt des menschlichen X Chromosoms betrachtet, der 10 Gene enthält und den *GenBank* [17] Bezeichner HUMFLNG6PD hat. Da in der DNA nur 4 verschiedene Zeichen (A,C,G,T) vorkommen, benötigt man nur $log_2(4) = 2$ Bits für ein Zeichen. Die korrigierte Dateigröße beträgt damit 54862 Bytes. Nach Kompression mit *gzip* ist die Datei 61567 Bytes lang, also um 6705 Bytes länger als vorher. Anstatt die Dateigröße zu reduzieren, vergrößert *gzip* die Datei um mehr als 12 %.

Texte, die Informationen enthalten, gelten als nicht zufällig und sollten komprimierbar sein. Diese DNA Sequenz enthält immerhin die Information über 10 Gene und sollte daher nicht als zufällig betrachtet werden. Da diese Sequenz aber nicht mit *gzip* komprimiert werden kann, stellt sich folgende Frage: *Sind DNA Sequenzen generell nicht komprimierbar oder sind die Kompressionsalgorithmen nicht in der Lage, DNA zu komprimieren?*

Diese Frage ist ein zentrales Thema der Diplomarbeit. Um die Anwort vorweg zu nehmen: DNA Sequenzen sind im Allgemeinen komprimierbar, auch wenn die Kompressionsraten sich nicht mit denen von Texten, die menschliche Sprache enthalten, vergleichen lassen, und es gibt Algorithmen, die DNA Sequenzen komprimieren können.

Diese Arbeit ist inhaltlich in zwei Teile aufgeteilt. Im ersten Teil werden DNA Sequenzen aus dem Blickwinkel der Informationstheorie betrachtet. Dabei soll untersucht werden, wodurch sich DNA Sequenzen von menschlicher Sprache unterscheiden und welche Charakteristiken ein guter DNA Kompressor ausnutzen sollte. Im Anschluß daran werden verschiedene DNA spezifische Kompressionsverfahren [36, 31, 24, 12, 37] erläutert und ihre Leistung verglichen.

Im zweiten Teil wird die Informationstheorie als Hilfsmittel für biologische Fragestellungen verwendet. Diese Fragestellungen betreffen Klassifikation, das Finden von signifikanten Strukturen in einer Sequenz sowie den Vergleich von Genomen. Weiterhin können mit informationstheoretischen Maßen bestimmte Regionen im Genom charakterisiert werden. So kann man zeigen, daß es einen Entropieunterschied zwischen kodierenden (Exons) und nichtkodierenden Regionen eines Gens (Introns) gibt. Auf die beiden zuletzt genannten Themen wird ausführlich eingegangen. Dabei werden die Arbeiten von Forschern [32, 21], die diese Methoden vorgeschlagen haben, aufgegriffen und die Ergebnisse anhand von neueren, größeren Datensätzen überprüft. Weiterhin soll untersucht werden, ob sich informationstheoretische Verfahren auch für andere biologische Fragestellungen verwenden lassen. Außerdem werden Gebiete gezeigt, wo sich die vorgestellten Methoden nur noch eingeschränkt anwenden lassen.

In Kapitel 2 und 3 werden zunächst biologische bzw. informationstheoretische Grundlagen erklärt. Da in der Arbeit zwei verschiedene Wissensgebiete, die Biologie und die Informationstheorie, behandelt werden, sollen in den zwei Kapiteln eine kurze Einführung gegeben und wichtige Begriffe definiert werden, die im späteren Verlauf eine Rolle spielen.

Kapitel 4 beschäftigt sich mit den hier schon kurz angesprochenen Themen, die genetische Sequenzen für die Informationstheorie und informationstheoretische Methoden für die Biologie interessant machen.

In Kapitel 5 werden die Grundlagen von Standardkompressoren erklärt. Dies ist wichtig, da die Funktionsweise der DNA spezifischen Kompressoren auf den hier erläuterten Prinzipien basiert. Es werden Unterschiede zwischen menschlichem Text und DNA Sequenzen aufgezeigt, die deutlich machen, warum Standardkompressoren keine oder nur eine geringe Kompression erreichen und worauf spezielle DNA Kompressionsverfahren achten sollten. Am Ende des Kapitels wird die Kompressionsrate verschiedener Kompressionsverfahren anhand von Benchmarksequenzen verglichen.

Kapitel 6 behandelt ausführlich die besten DNA Kompressionsverfahren, die zur Zeit verfügbar sind. Dabei wird zuerst auf Entropieschätzer eingegangen, die keine komprimierte Datei erzeugen, sondern nur die erreichbare Kompressionsrate schätzen. Anschließend werden echte Kompressoren vorgestellt. Ein Vergleich der Methoden anhand der in Kapitel 5 verwendeten Sequenzen zeigt, daß eine signifikant höhere Kompressionsrate erreicht werden kann. Soweit bekannt, werden außerdem Aussagen über die Laufzeit und Ressourcenanforderung der Programme gemacht.

Kapitel 7 beschäfigt sich mit der Anwendung der Kompression beim Vergleich von Genomen. Aus den Distanzdaten, die ein paarweiser Genomvergleich liefert, ist es möglich, einen Stammbaum der verwendeten Organismen zu rekonstruieren. Zum Verständnis ist es notwendig, sich kurz mit Distanzdaten und phylogenetischen Bäumen zu beschäftigen. Anschließend wird gezeigt, wie aus der bedingten Kompression mit Hilfe der Kolmogorov Komplexität ein Distanzmaß gebildet werden kann. Dieses Distanzmaß wird benutzt, um auf der Grundlage der Genome von 41 Bakterien einen Stammbaum zu rekonstruieren. Zum Schluß werden anhand dieses Beispiels die Vor– und Nachteile dieser Methode diskutiert.

In Kapitel 8 geht es um die angesprochenen Entropieunterschiede der Exons und Introns. Die früheren Arbeiten zu diesem Thema werden zuerst vorgestellt und danach die Experimente an größeren Datensätzen wiederholt. Die Resultate konnten dabei bestätigt werden und lassen sich für ein breites Spektrum von Organismen verallgemeinern. Im Anschluß daran wird geprüft, wie gut sich diese

charakteristischen Unterschiede bei der Vorhersage von Genen verwenden lassen.

Die Grenzen zwischen Exons und Introns werden Spleisstellen genannt (siehe Kap. 2.2). Für viele Anwendungen ist es wichtig, in einer Gensequenz alle Spleisstellen zu identifizieren. Dafür ist es nötig, daß die Charakteristika der Spleisstellen bekannt sind. Kapitel 9 beschäfigt sich mit der Frage, welche Positionen in der Umgebung einer Spleisstelle einen Beitrag zur korrekten Erkennung in einer Zelle leisten. Nach der Vorstellung bisheriger Arbeiten soll diese Frage mit Hilfe der bedingten Entropie beanwortet werden, wobei die Datensätze aus Kapitel 8 verwendet werden.

Zuletzt werden die Ergebnisse dieser Diplomarbeit in Kapitel 10 zusammengefaßt, ungeklärte Fragen angesprochen und Themen diskutiert, in denen weitere Untersuchungen interessant sind.

Kapitel 2

Biologische Grundlagen

In diesem Kapitel werden biologische Grundlagen erklärt, die für das Verständnis der folgenden Kapitel notwendig sind. Es werden die wichtigsten biologischen Moleküle vorgestellt und kurz auf die Genexpression sowie die Mechanismen der Evolution eingegangen. Der Inhalt stammt im Wesentlichen aus [55], wo die hier nur kurz angeschnittenen Themen viel detaillierter beschrieben sind.

2.1 Die genetischen Moleküle

2.1.1 DNA

Die grundlegenden Bausteine der DNA (desoxyribonucleic acid) sind Desoxynukleotide. Als Desoxynukleotid bezeichnet man das Monomer einer Nukleinsäure, das an einen ringförmigen Zucker mit 5 Kohlenstoffatomen, die Desoxyribose, gebunden ist. Die 5 C – Atome der Desoxyribose sind durchnummeriert. Mit Hilfe von Phosphodiesterbrücken zwischen dem 3’ – Kohlenstoffatom der Desoxyribose eines Nukleotids und dem 5’ – Kohlenstoffatom der Desoxyribose eines benachbarten Nukleotids kommt es zur Bildung langer Nukleotidketten (Abb. 2.1). Jede Kette besitzt eine freie Phosphatgruppe an einem 5’ C – Atom und eine freie Hydroxylgruppe (OH) am 3’ C – Atom am anderen Ende. Dadurch bekommt ein DNA–Strang eine Leserichtung. Gewöhnlich schreibt man eine Nukleotidkette vom 5’ zum 3’ Ende auf.

In der DNA kommen 4 verschiedene Basen vor: Adenin (A), Cytosin (C), Guanin (G) und Thymin (T). Zwei komplementäre Basen können sich über Wasserstoffbrücken miteinander verbinden. In der DNA binden sich immer Adenin und

Thymin sowie Cytosin und Guanin. Diese Bindung kommt mit sehr hoher Spezifität zustande. Zwei DNA–Einzelstränge bilden aufgrund der komplementären Basenpaarung einen Doppelstrang. Dabei sind die beiden Einzelstränge antiparallel, d.h. der Sinnstrang verläuft in 5' $\rightarrow$ 3' Richtung und der Antisinnstrang in 3' $\rightarrow$ 5' Richtung. Die Reihenfolge der Basen eines Stranges bestimmt vollständig die Basensequenz des anderen.

Die dreidimensionale Struktur der DNA wurde 1953 von Watson und Crick aufgeklärt. Nach ihrem bekannten Strickleitermodell bilden DNA–Doppelstränge eine gewundene Helix, wobei die Basen in das Innere der Spirale hineinragen (Abb. 2.2). Diese Helix hat einen konstanten Durchmesser, was durch die fast identische räumliche Größe zweier komplementärer Basen (A – T, C – G) erklärt werden kann. Andere Paarungen würden zu einer instabilen Helix mit wechselndem Durchmesser führen. Die gewundene Spiralstruktur verleiht der DNA eine hohe Stabilität und Widerstandsfähigkeit gegenüber DNA spaltenden Enzymen. Normalerweise liegt die DNA in der sogenannten B – Form vor, die sich durch einen Drehwinkel von 36° und einer Zahl von 10 Basenpaaren pro Helixumdrehung auszeichnet. Unter bestimmten Bedingungen kann die DNA noch zwei weitere Helixtypen bilden, die A – DNA und Z – DNA, die sich durch veränderte Winkel und Abstände zwischen den Nukleotiden unterscheiden. Dabei ist die A – Form etwas kürzer und dicker, die Z – Form länger und dünner als die B – DNA (Abb. 2.3).

2.1.2 RNA

Neben der DNA gibt es eine weitere Nukleinsäure, die in einer Zelle vorkommt, die RNA (ribonucleic acid). Die RNA ist chemisch weitgehend ähnlich zur DNA mit zwei Ausnahmen. Statt Thymin (T) kommt in der RNA Uracil (U) vor. Das Zuckergrundgerüst besteht aus Ribose, was eine geringere Resistenz des Moleküls für spaltende Enzyme zur Folge hat. Im Gegensatz zu DNA liegt die RNA als Einzelstrang vor und ist wesentlich kürzer als ein DNA–Doppelstrang. RNA's spielen unter anderem eine wichtige Rolle bei der Genexpression.

2.1.3 Proteine

Jede Zelle besteht zum großen Teil aus Proteinen, die für den strukturellen Aufbau der Zelle verantwortlich sind und eine Schlüsselrolle bei fast allen intra- und interzellulären Prozessen besitzen. Bausteine der Proteine sind 20 verschiedene Aminosäuren. Man unterscheidet die Primär–, Sekundär– und Tertiärstruktur der Ei-

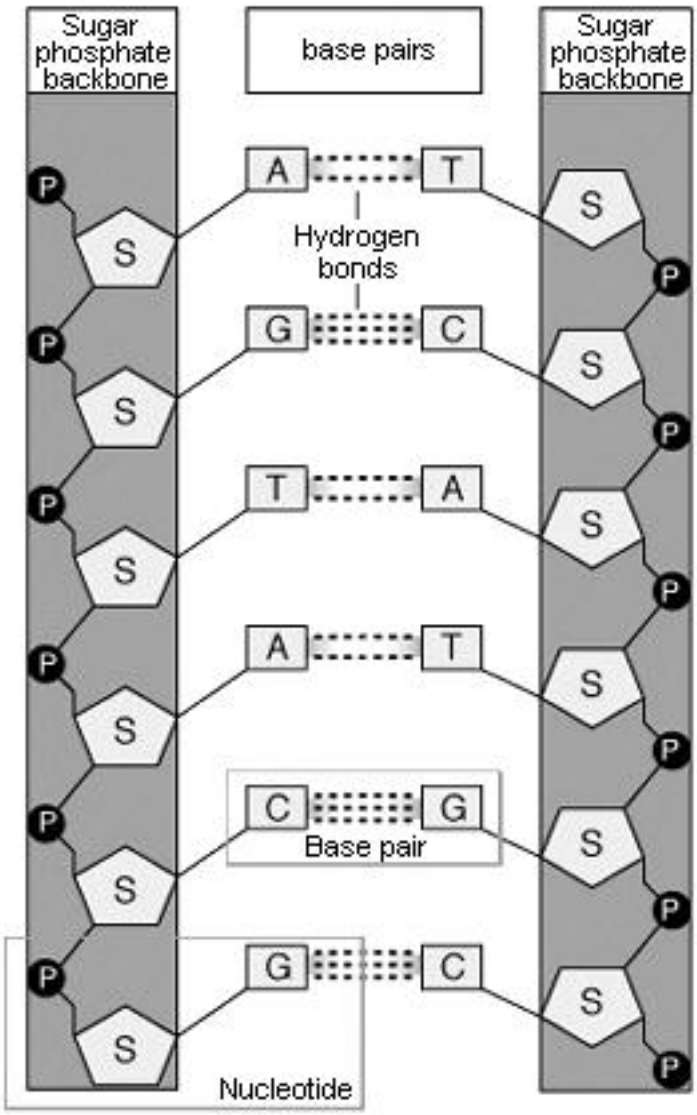


Abbildung 2.1: DNA–Doppelstrang [59]

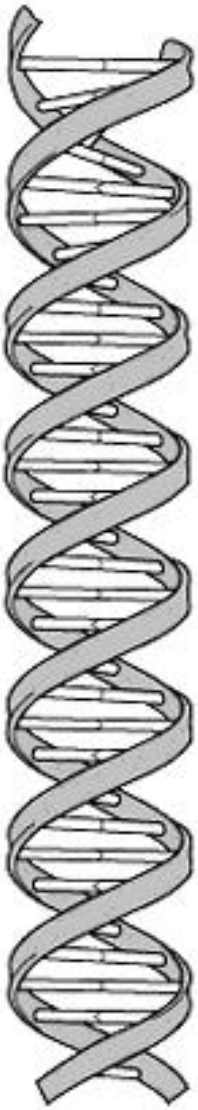
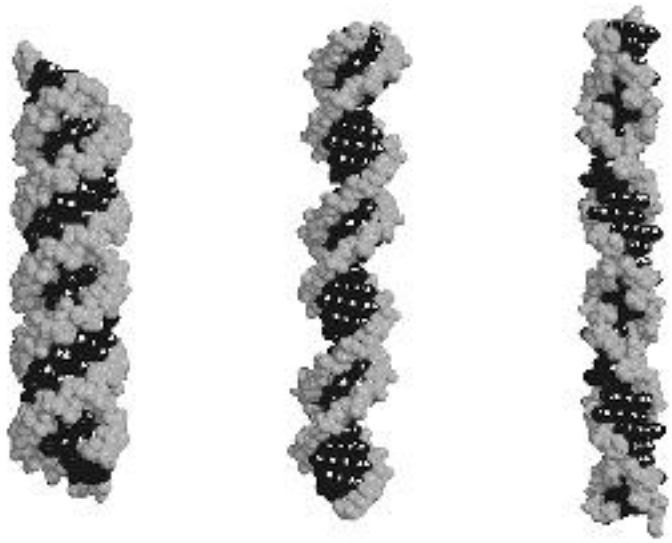


Abbildung 2.2: DNA Helix [59]



A – DNA B – DNA Z – DNA

Abbildung 2.3: Dreidimensionale Strukturen der DNA [44]

weiße. Als Primärstruktur bezeichnet man die lineare Abfolge der Aminosäuren, die mit Hilfe einer Peptidbindung lange Polypeptidketten bilden. Polypeptidketten falten sich auf charakteristische Weise, wodurch die Sekundärstruktur entsteht, die maßgeblich durch die Reihenfolge der Aminosäuren bestimmt wird. Die meisten Proteine, insbesondere Enzyme, bilden wiederum komplexe, dicht gefaltete, dreidimensionale Strukturen, die nötig sind, um die spezifische Enzymaktivität zu ermöglichen. Diese Tertiärstruktur entsteht durch Wechselwirkungen zwischen Aminosäuren an unterschiedlichen Stellen der Polypeptidkette und hängt sowohl von der Sekundärstruktur, als auch von den chemischen Eigenschaften der Umgebung ab. Manche Proteine sind aus mehreren Polypeptidketten aufgebaut (Beispiel Hämoglobin), wodurch sich eine Quartärstruktur ergibt.

2.2 Genexpression

DNA ist der Träger der Erbinformation eines Organismus. Die DNA kodiert dabei nicht nur die gesamte Information für den strukturellen Aufbau des Lebewesens, sondern auch alle Mechanismen für die Erhaltung und Replikation der DNA in den Zellen.

Obwohl die DNA in einer dreidimensionalen Struktur vorliegt, ist die enthaltene Information als lineare Abfolge der Basen darstellbar. Da der komplementäre Strang keine zusätzliche Information liefert, genügt es, wenn man sich auf die Darstellung eines Strangs bezieht.

Die Basensequenz der DNA bestimmt allein die Primärstruktur eines Proteins. Der Weg von der DNA zum synthetisierten Protein besteht im Wesentlichen aus zwei Schritten (Abb. 2.4).

1. Bestimmte Teile eines DNA–Strangs werden durch spezielle Enzyme (RNA Polymerase) abgelesen, was als *Transkription* bezeichnet wird. Dabei entsteht *Messenger–RNA* (*mRNA*) als exakte, komplementäre Kopie eines Teilstücks eines Stranges.

2. Aus der mRNA werden mit Hilfe von Ribosomen Proteine synthetisiert (*Translation*).

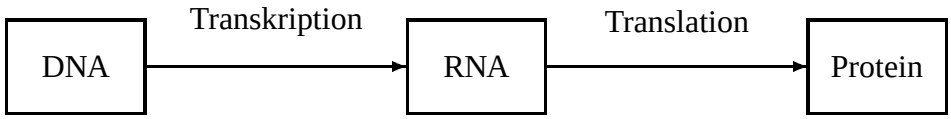


Abbildung 2.4: Fluß der genetischen Information

Die *Genexpression* beginnt mit der Transkription eines DNA–Strangs. Dabei unterscheidet man den informationstragenden *Sinnstrang* und den komplementären *Antisinnstrang*, der für das Ablesen benutzt wird. Dadurch wird sichergestellt, daß die mRNA, die komplementär zum Antisinnstrang ist, genau die Basenabfolge des Sinnstrangs enthält (Abb. 2.5).

Das Alphabet der DNA besteht aus 4 Buchstaben, A,C,G und T, während das Proteinalphabet 20 Buchstaben hat. Daher sind 3 aufeinanderfolgende Basen (Triplet oder Codon) nötig, um eindeutig eine Aminosäure zu bestimmen. Mit einem Triplet hat man $4^3 = 64$ eindeutige Zuordnungen, es gibt aber nur 20 Aminosäuren. Da alle möglichen Codons auch verwendet werden, ist der *genetische Kode* zwar eindeutig, aber degeneriert. Zudem ist der Kode universell und damit bei allen Organismen nahezu identisch. Aus der Kodetabelle (Tabelle 2.1) wird ersichtlich, daß oftmals eine Aminosäure schon durch 2 Basen bestimmt wird und die dritte redundant ist. 3 Codons dienen als Stoppsignal und stellen das Ende eines kodierenden Abschnittes dar. Die restlichen 61 Codons kodieren Aminosäuren,

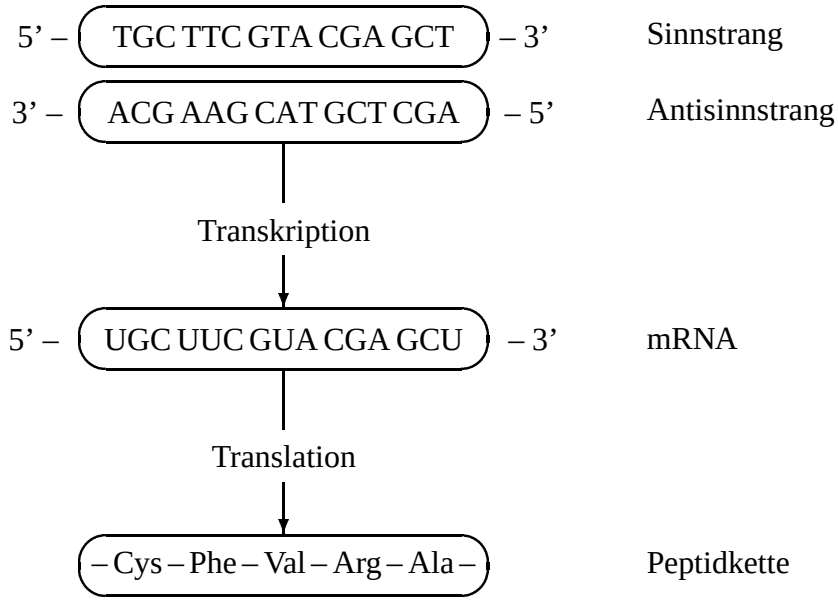


Abbildung 2.5: Schema der Transkription und Translation

wobei das Triplet *ATG* zur Kennzeichnung des Beginns eines kodierenden Abschnittes benutzt wird und gleichzeitig die Aminosäure Methionin kodiert. Ein kodierender Abschnitt oder *Gen* ist eine Abfolge von Codons mit einem bestimmten *Leseraster*, das angibt, an welcher Position das erste Codon beginnt (Abb. 2.6).

Bei höheren Lebewesen ist die DNA einer Zelle im Zellkern enthalten, der durch eine Membran vom Zellplasma abgetrennt ist. Diese Organismen bezeichnet man als *Eukaryoten*. Bei vielen einfacheren Lebewesen, wie Bakterien und

TGT	ACT	GAA	CGT	ACG	AGT	TAC	CCG	TAG	GTG	C
T	GTA	CTG	AAC	GTA	CGA	GTT	ACC	CGT	AGG	TGC
TG	TAC	TGA	ACG	TAC	GAG	TTA	CCC	GTA	GGT	GC

Abbildung 2.6: Die drei möglichen Leseraster

Ala	**Arg**	**Asp**	**Asn**	**Cys**	**Glu**	**Gln**	**Gly**	**His**	**Ile**	**Met**
GCC	CGC	GAC	AAC	TGC	GAG	CAG	GGC	CAC	ATC	ATG
GCT	CGT	GAT	AAT	TGT	GAA	CAA	GGT	CAT	ATT	
GCG	CGG						GGG		ATA	
GCA	CGA						GGA			
	AGG									
	AGA									
Leu	**Lys**	**Phe**	**Pro**	**Ser**	**Thr**	**Trp**	**Tyr**	**Val**		**Stop**
CTC	AAG	TTC	CCC	TCC	ACC	TGG	TAC	GTC		TGA
CTT	AAA	TTT	CCT	TCT	ACT		TAT	GTT		TAG
CTG			CCG	TCG	ACG			GTG		TAA
CTA			CCA	TCA	ACA			GTA		
TTG				AGC						
TTA				AGT						

Tabelle 2.1: Der genetische Kode

Viren, liegen die DNA–Stränge im Zellplasma; einen Zellkern gibt es nicht. Diese Gruppe von Organismen bezeichnet man als *Prokaryoten*. Bei Eukaryoten enthalten außerdem Zellorganellen (Mitochondrien, Chloroplasten) eigene DNA Genome. Es gibt zwischen Pro– und Eukaryoten wesentliche, strukturelle Unterschiede der DNA, auf die im Folgenden kurz eingegangen werden soll.

Das Genom eines Lebewesens kann grob in zwei funktionell unterschiedliche Bereiche aufgeteilt werden. *Kodierende Bereiche* enthalten die Information über die Aminosäuresequenz eines Proteins. *Nichtkodierende Bereiche* enthalten u.a. Regulationsregionen, die die Intensität der Genexpression steuern. Ob die restlichen nichtkodierenden Bereiche eine biologische Funktion haben oder überflüssige DNA sind, ist noch weitgehend ungeklärt. Im Gegensatz zu Prokaryoten bestehen die Genome der Eukaryoten zum größten Teil aus nichtkodierenden Regionen. Zum Beispiel besteht Schätzungen zufolge das menschliche Genom aus 95% nichtkodierenden Regionen (Quelle: Nature Vol. 409).

Aber auch Gene enthalten Stücke, die keine Aminosäuren kodieren. Man bezeichnet diese als *Introns*, die kodierenden Abschnitte innerhalb eines Gens als *Exons* (Abb. 2.7). Die transkribierte mRNA enthält eine genaue Kopie aller Exons und Introns des Gens. Bevor die mRNA in eine Aminosäuresequenz translatiert wird, werden alle Introns entfernt, ein Vorgang, den man *Spleissen* nennt.

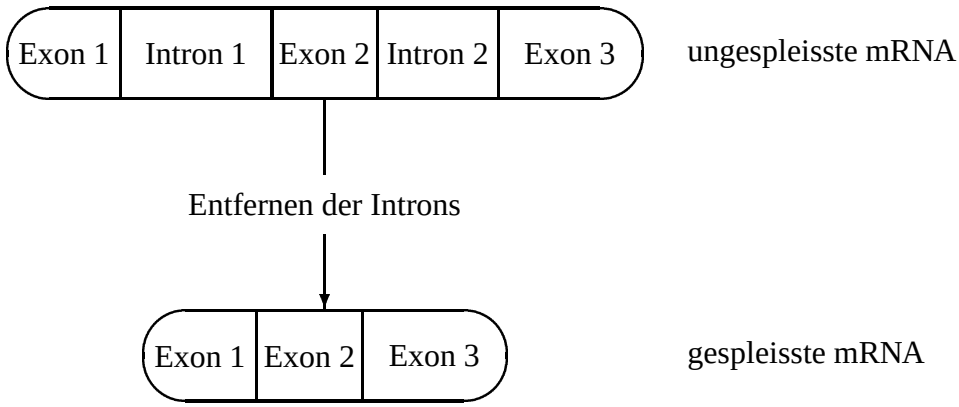


Abbildung 2.7: Spleissen der mRNA

Während dieses sehr spezifischen Prozesses werden die genauen Grenzen zwischen Exon und Intron (*Spleisstellen*) erkannt, die mRNA an diesen Stellen gespalten, das Intron herausgeschnitten und die zwei benachbarten Exons miteinander verbunden. Die gespleisste (reife) mRNA besteht aus einer Folge von Exons und kann nun in die entsprechende Aminosäuresequenz übersetzt werden.

Ein wesentlicher Unterschied zwischen Pro– und Eukaryoten ist, daß Introns in der Regel nur bei Eukaryoten vorkommen. Bei Prokaryoten besteht ein Gen also nur aus einem zusammenhängenden Exon.

2.3 Mechanismen der Evolution

Die genetische Information ist ständigen Veränderungen unterworfen, was die Anpassung und Weiterentwicklung von Organismen ermöglicht. Diese Veränderungen können auf der Ebene der Nukleotide geschehen oder ganze Teile von DNA–Strängen und Chromosomen betreffen.

Die einfachste Veränderung ist das Einfügen (Insertion), Löschen (Deletion) und Austauschen von Basen an beliebigen Positionen eines DNA–Stranges, was zufällig passiert und durch äußere Einflüsse, wie Hitze–, Kälteschocks, ionisierende Strahlung und chemische Substanzen verstärkt werden kann.

Neben Veränderungen an einzelnen Basen können auch ganze DNA Regionen entfernt oder kopiert werden. So sind häufig benutzte Gene oftmals mehrfach im

Genom vorhanden, um dadurch eine parallele Transkription zu ermöglichen (Genamplifikation). Die kopierten Bereiche können entweder an benachbarten Positionen oder an weit entfernten Stellen im Genom eingefügt werden. Wenn eine Kopie immer wieder hintereinander eingefügt wird, entsteht eine *Tandemwiederholung*. Tatsächlich bestehen große Teile des menschlichen Genoms aus solchen Tandem Repeats, deren zugrunde liegendes Muster sich viele tausend mal wiederholt.

Aufgrund der entgegengesetzten Richtungen der beiden DNA–Stränge, kann ein Teilstück eines Stranges in umgekehrter Reihenfolge in den komplementären Strang eingefügt werden. Das ergibt eine *umgekehrte, komplementäre Wiederholung*, auch Palindrom genannt. Beispiele für diese Wiederholungen sind in Abbildung 2.8 gezeigt (N steht für eine beliebige Base).

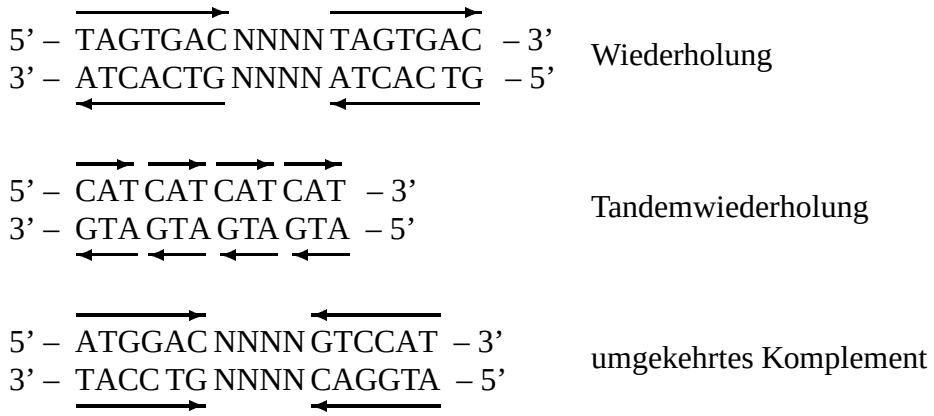


Abbildung 2.8: Wiederholungen in DNA Sequenzen

Kapitel 3

Informationstheoretische Grundlagen

Im folgenden Kapitel werden die Begriffe Entropie und Kompression sowie ihr Zusammenhang erklärt. Shannon [54] erkannte zuerst die Bedeutung der Entropie und schuf damit die Grundlagen der Informationstheorie. Entropie und Kompression sind zentrale Begriffe dieser Arbeit.

3.1 Entropie

Gegeben sei ein Alphabet $X = \{x_1, \ldots, x_n\}$ mit n Zeichen und eine Wahrscheinlichkeitsverteilung $P = (p(x_1), \ldots, p(x_n))$, die jedem Zeichen aus X eine Wahrscheinlichkeit zuordnet. Da P eine Wahrscheinlichkeitsverteilung ist, gilt

$$p(x_i) \geq 0 \ \ \forall i \quad \text{und} \quad \sum_{i=1}^{n} p(x_i) = 1.$$

Definition 3.1 (Quelle) *Als Quelle bezeichnet man den Zufallsprozeß, der mit der Wahrscheinlichkeitsverteilung P nacheinander einzelne Zeichen aus dem Alphabet X ausgibt.*

Das Münzwurfexperiment ist ein einfaches Beispiel für eine Quelle. Hier ist $X = \{K, Z\}$ und $P = (0.5, 0.5)$. In jedem Wurf wird genau ein Zeichen (K = Kopf, Z = Zahl) mit gleicher Wahrscheinlichkeit generiert.

Definition 3.2 (Entropie) *Sei eine Quelle (X, P) gegeben. Die binäre Entropie einer Quelle ist definiert als*

$$H(P) = -\sum_{i=1}^{n} p(x_i)\ log_2(p(x_i)),$$

wobei $0\ log_2(0) = 0$ gesetzt wird.

Im Folgenden wird unter Entropie immer die binäre Entropie verstanden. Für die Entropie spielt allein die Wahrscheinlichkeitsverteilung P eine Rolle; welche Zeichen die Quelle konkret generiert, ist bei ihrer Berechnung irrelevant. Die Entropie gibt die Unsicherheit über das Zeichen an, das die Quelle als nächstes generiert. Das ist äquivalent zu dem Informationsgewinn, den man erhält, wenn das nächste Zeichen beobachtet wird. Je höher also die Entropie, um so zufälliger werden Zeichen generiert.

Es gilt: $H(P) \geq 0$ und $H(P) \leq log_2(n)$. Die Entropie ist genau dann 0, wenn ein Zeichen mit Wahrscheinlichkeit 1 generiert wird und die anderen Symbole Wahrscheinlichkeit 0 haben. In diesem Fall gibt es keine Unsicherheit über das folgende Zeichen, da bekannt ist, was die Quelle als nächstes generiert. Wenn P eine Gleichverteilung ist, wird $H(P)$ maximal, da die Unsicherheit über das folgende Zeichen ebenfalls maximal ist. Im Beispiel des Münzwurfes ist die Entropie $-2\ (0.5\ log_2(0.5)) = 1 = log_2(2)$, da beide Ausgänge des Zufallsexperimentes mit gleicher Wahrscheinlichkeit auftreten. Wenn die Münze nicht fair ist und z.B. zu 90% Kopf fällt, ist die Entropie nur noch $-0.1\ log_2(0.1) - 0.9\ log_2(0.9) = 0.47$, da sich die Unsicherheit über den Ausgang eines Wurfes stark verringert hat. Der Verlauf der Entropiefunktion für ein Zufallsexperiment mit 2 Ausgängen ist in Abbildung 3.1 gezeigt.

3.2 Kodierung und Kompression

Um die von einer Quelle generierten Zeichen in einer binären Datei abzuspeichern, benötigt man für jedes Zeichen des Alphabets eine Zuordnung zu einem binären Kodewort.

Definition 3.3 (Kode und Kodierung) *Ein binärer Kode C ist eine nicht leere Teilmenge aus der Menge aller Wörter über dem Alphabet $\{0,1\}$. Kodewörter sind die Elemente von C. Eine Kodierung (C, f) besteht aus einem Kode C und einer injektiven Funktion $f : X \rightarrow C$, die jedem Zeichen aus X ein eindeutiges Kodewort aus C zuordnet.*

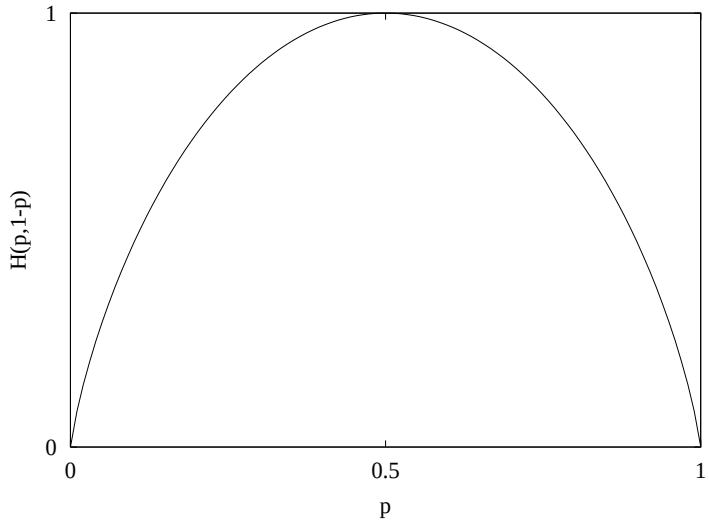


Abbildung 3.1: Verlauf der Entropiefunktion für $P = (p, 1-p)$

Definition 3.4 (mittlere Kodewortlänge) *Die mittlere Kodewortlänge einer Kodierung gibt an, wieviel Bits im Durchschnitt gebraucht werden, um ein Zeichen zu kodieren und ist definiert als*

$$AveLen(C, f) = \sum_{i=1}^{n} p(x_i)\ |f(x_i)|,$$

wobei $|f(x_i)|$ die Länge des Kodewortes für x_i in Bits ist.

Man interessiert sich nur für Kodierungen, die einen eindeutig dekodierbaren Kode liefern.

Definition 3.5 (eindeutig dekodierbarer Kode) *Ein Kode ist eindeutig dekodierbar, wenn kein Kodewort ein Präfix eines anderen Kodewortes ist.*

Bsp.: Sei $X = \{a, b, c, d\}$ und $P = (0.5, 0.3, 0.1, 0.1)$. Der Kode $a \rightarrow 0$, $b \rightarrow 01$, $c \rightarrow 10$, $d \rightarrow 11$ ist nicht eindeutig dekodierbar, da der binäre String 0010 als “aac” oder “aba” dekodiert werden kann. Der Kode $a \rightarrow 000$, $b \rightarrow 01$, $c \rightarrow 10$, $d \rightarrow 11$ hingegen ist eindeutig dekodierbar, da kein Kodewort der Präfix eines anderen ist. Seine mittlere Kodewortlänge ist 2.5.

Von allen eindeutig dekodierbaren Kodes möchte man den benutzen, der die mittlere Kodewortlänge minimiert. Mit k Bits kann man 2^k verschiedene Zahlen speichern. Eine *triviale Kodierung* für X benutzt daher n Kodewörter, die alle eine feste Länge von $c = \lceil log_2(n) \rceil$ Bits haben. Wenn die Wahrscheinlichkeit für alle Zeichen gleich ist, ist dieser Kode optimal, und die mittlere Kodewortlänge ist c Bits. Das heißt, für jedes Alphabet X und jede Wahrscheinlichkeitsverteilung P gibt es einen Kode, dessen mittlere Kodewortlänge c Bits ist. Der Kode im obigen Beispiel ist ineffizient, da seine mittlere Kodewortlänge größer als $log_2(4) = 2$ Bits ist.

Falls einige Zeichen häufiger auftreten als andere, gibt es effizientere Kodes mit kürzerer mittlerer Kodewortlänge. Im obigen Beispiel werden a und b viel häufiger als c und d ausgegeben. Wenn man den Zeichen mit hoher Wahrscheinlichkeit kurze Kodewörter und den seltenen Zeichen lange Kodewörter zuordnet, erreicht man eine kürzere mittlere Kodewortlänge. So hätte der Kode $a \rightarrow 0$, $b \rightarrow 10$, $c \rightarrow 110$, $d \rightarrow 111$ eine mittlere Kodewortlänge von 1.7 Bits und wäre trotzdem eindeutig dekodierbar.

Der effizienteste Kode ordnet jedem Zeichen ein Kodewort der Länge $-log_2(p(x_i))$ zu. Dann ergibt sich als mittlere Kodewortlänge gerade die Entropie der Quelle. Das Noiseless Coding Theorem von Shannon [54] beweist, daß die mittlere Kodewortlänge immer größer oder höchstens gleich der Entropie einer Quelle ist. Damit kann die Entropie als untere Schranke der mittleren Kodewortlänge angesehen werden.

Theorem 3.1 (Noiseless Coding Theorem) *Sei (C, f) eine eindeutig dekodierbare Kodierung einer Quelle (X, P). Dann gilt $H(P) \leq AveLen(C, f)$ mit Gleichheit g.d.w. $-log_2(p(x_i)) = |f(x_i)| \quad \forall\, i = 1, \ldots, n$.*
Weiterhin gibt es für jede Quelle (X, P) eine eindeutig dekodierbare Kodierung (C, f), die $AveLen(C, f) < H(P) + 1$ erfüllt.

Die zweite Aussage des Theorems zeigt, daß die Entropie auch eine obere Schranke liefert.

Eine Datei, die eine Sequenz von m Zeichen speichert, benötigt $m * c$ Bits, wenn die triviale Kodierung verwendet wird. Ein verlustfreier Kompressor versucht den exakten Informationsgehalt der Datei in weniger als $m * c$ Bits zu speichern. Kompression gelingt immer dann, wenn bestimmte Regelmäßigkeiten, wie Wiederholungen und Abhängigkeiten von den vorherigen Zeichen, oder Unterschiede in der Häufigkeit der einzelnen Zeichen ausgenutzt werden können. Falls

die Quelle Zeichen gleichverteilt auswählt, ist die Entropie maximal und die Datei könnte nicht komprimiert werden. Das kann als Qualitätstest für einen Zufallsgenerator dienen. So würde sich die Ausgabe eines perfekten Zufallsgenerators nicht komprimieren lassen. Falls dies trotzdem gelingt, ist der Zufallsgenerator nicht perfekt.

Um die Güte der Kompression zu beurteilen, kann man die *durchschnittliche Kodewortlänge* berechnen, indem die Bitzahl der komprimierten Datei $|O|$ durch die Anzahl der gespeicherten Zeichen geteilt wird. Ein alternatives Maß ist die *Kompressionsrate*, die durch $1 - |O|/|I|$ berechnet wird, wobei $|I|$ die Bitzahl der Originaldatei ist. Je höher die Kompressionsrate, desto kleiner ist die durchschnittliche Kodewortlänge und damit die Dateigröße. Wenn die Kompressionsrate 0 ist, hat keine Kompression stattgefunden, bei einer negativen Kompressionsrate hat der Kompressor mehr Bits als die triviale Kodierung benötigt.

Kapitel 4

Motivation für die Informationstheorie in der Biologie

Hier soll erläutert werden, warum biologische Sequenzen für die Informationstheorie interessant sind. Desweiteren werden Bereiche genannt, in denen informationstheoretische Anwendungen in der Biologie eine Rolle spielen. Auf einige dieser Bereiche wird in späteren Kapiteln detailliert eingegangen.

4.1 Informationstheoretische Fragestellungen

Der strukturelle Aufbau eines Organismus hat wenig mit Zufälligkeit oder Chaos zu tun. Da die Information über Aufbau und Funktion eines Lebewesens in der DNA kodiert ist, erwartet man gleiches für eine gegebene DNA Sequenz. In den folgenden Kapiteln wird DNA als Text über dem Alphabet (A, C, G, T) betrachtet. Wenn eine Sequenz nicht zufällig generiert wurde, sollte sie komprimierbar sein. Tatsächlich gibt es auch biologische Hinweise, die Redundanz und statistische Verzerrungen der DNA aufzeigen, was eine Kompression ermöglichen sollte. Die DNA ist ein geordnetes Molekül mit Strukturelementen wie Wiederholungen (Tandem Repeats, umgekehrte Komplemente), duplizierten Genen und CpG Inseln (Bereiche im Genom, in denen fast ausschließlich C und G vorkommt).

Vergleichbar mit menschlicher Sprache (beispielsweise englischen Text), die sowohl Redundanz als auch eine strenge Ordnung aufweist, sollten Kompressionsverfahren wie LZ77 (Kap. 5.1.1) oder LZ78 (Kap. 5.1.2) gute Ergebnisse bei DNA Sequenzen liefern. Überraschenderweise gelingt es diesen Kompressoren aber nicht, DNA zu komprimieren (Kap. 5.4), obwohl sie bei englischem Text sehr

hohe Kompressionsraten erreichen. Trotzdem impliziert die offenbar nur schwer zu erreichende Kompression nicht, daß DNA in Wirklichkeit doch vollkommen zufällig ist. Dies würde bedeuten, daß jede zufällig erzeugte DNA Sequenz zu einem lebensfähigen Organismus führen würde. Aus der Biologie weiß man, daß ein Teil der Mutationen zum Tod des Lebewesens führen kann. Also muß es auch in DNA Sequenzen gewisse Bedingungen geben, die eine in der Natur vorhandene Sequenz erfüllen muß, auch wenn diese Bedingungen weniger offensichtlich sind als bei englischem Text.

Die schwer erreichbare Kompression macht DNA Sequenzen zu einer neuen Herausforderung für die Informationstheorie [37]. Darüber hinaus ist man an einem genauen Modell der DNA generierenden Quelle interessiert. Ein Kompressionsalgorithmus kann eine Sequenz nur komprimieren, wenn er charakteristische Eigenschaften der Quelle ausnutzt. Je höher die resultierende Kompressionsrate, desto wahrscheinlicher besitzt die Quelle diese Charakteristiken. Dadurch erlaubt die Verwendung eines Kompressionsverfahrens die Modellierung der zugrundeliegenden Quelle [46]. Offenbar sind die Eigenschaften der DNA Quelle verschieden zu einer Quelle, die menschliche Sprache erzeugt.

Oftmals ist es nicht möglich, die Entropie einer Quelle exakt zu bestimmen, da man eine unendlich lange Zeichenfolge abwarten müßte. Allerdings kann die Entropie der Quelle aus der Entropie einer ausreichend langen Sequenz geschätzt werden. Da die durchschnittliche Kodewortlänge eines Kompressors immer größer oder höchstens gleich der Entropie ist (Noiseless Coding Theorem, Kap. 3.2), eignen sich Kompressionsverfahren als Schätzer für die Entropie einer Quelle. Die durchschnittliche Kodewortlänge ist dann eine obere Schranke der Entropie. Mit der Entropie von DNA Sequenzen kann man auf den Gehalt an Information Rückschlüsse ziehen.

Es sei bemerkt, daß spezielle DNA Kompressionsverfahren, die mittlerweile entwickelt worden sind (Kap. 6), nicht angewendet werden, um Speicherplatz in biologischen Sequenzdatenbanken wie *GenBank* [17] oder *EMBL* [16] zu sparen. Der Grund hierfür liegt wahrscheinlich in der viel zu geringen Geschwindigkeit existierender Algorithmen und der immer noch sehr niedrigen Kompressionsrate.

4.2 Anwendungen der Informationstheorie in der Biologie

Entropieschätzung kann man benutzen, um unterschiedliche Regionen der DNA zu charakterisieren, was von Farach et al. [21] für Exons und Introns gemacht

wurde. Zufällige Mutationen sollten in Exon– und Intronabschnitten gleichhäufig vorkommen. In Exons werden aber nur sinnvolle Mutationen, also Mutationen, die ein Weiterleben des Organismus nicht verhindern, erhalten bleiben. Da Introns keine Informationen über das Protein enthalten, erwartet man, daß in diesen Abschnitten beliebige Mutationen erlaubt sind. Dadurch müßte sich ein Entropieunterschied zwischen Exons und Introns ergeben, was von Farach et al. [21] nachgewiesen werden konnte. Unterschiede in der Entropie können zur Charakterisierung von Exons und Introns benutzt werden. In Kapitel 8 wird ausführlich auf dieses Thema eingegangen. Die Entropie als charakteristische Eigenschaft ist dabei unabhängig vom konkreten Inhalt der Sequenz. Man vergleicht also nicht eine Sequenz bezüglich bekannter Inhalte, sondern benutzt ausschließlich ihre statistischen Eigenschaften.

Kurze DNA Stücke (Oligonukleotide) können drei verschiedene dreidimensionale Formen einnehmen, die als A –, B – und Z – DNA bezeichnet werden. Loewenstern [34] hat gezeigt, daß ein Kompressionsalgorithmus mittels drei verschiedener Trainingsmengen, die bekannte Oligonukleotide der A –, B – bzw. Z– Form beinhalten, eine genaue Vorhersage für ein Oligonukleotid mit unbekannter 3D Struktur erstellen kann. Dieses Thema ist in [34] ausführlich beschrieben und soll hier nur kurz erwähnt werden. Der Kern der Methode basiert auf der Schätzung der bedingten Entropie (Definition in Kapitel 9.1) zwischen dem unbekannten Oligonukleotid und jeder der drei Trainingsmengen. Zur Schätzung der bedingten Entropie wird *CDNA* (Kap. 6.1.1) benutzt. Wenn die bedingte Entropie einen hohen Wert annimmt, gibt es wenig Ähnlichkeiten zwischen der Trainingsmenge und dem zu analysierenden Oligonukleotid. Andernfalls kann man auf gemeinsame Eigenschaften zwischen dem Oligonukleotid und der Trainingsmenge schließen. Aus der geschätzten bedingten Entropie kann man die Wahrscheinlichkeit berechnen, daß dieses Oligonukleotid dieselbe 3D Struktur hat wie die Elemente der Trainingsmenge. Mit Hilfe dieser Wahrscheinlichkeit und der Bayes Regel wird das unbekannte Oligonukleotid klassifiziert. Loewenstern erreichte mit dieser Methode eine Genauigkeit von 96.4 %.

Ein Kompressor versucht in der Eingabesequenz Regelmäßigkeiten zu finden, deren Beschreibung kürzer als die ursprüngliche Darstellung ist. Dadurch ermöglichen Kompressionsverfahren die Entdeckung von enthaltenen Strukturen einer Sequenz und beurteilen gleichzeitig deren Signifikanz [38]. So läßt sich eine Sequenz nur komprimieren, wenn sie signifikante Strukturen enthält. Dabei ist die Kompressionsrate ein Maß für die Signifikanz. Signifikante Regelmäßigkeiten in DNA Sequenzen sollten durch Mechanismen der Evolution entstanden sein und könnten eine biologische Bedeutung haben. Nicht signifikante Regelmäßigkeiten

sind wahrscheinlich ein Produkt des Zufalls und dürften keine biologische Rolle spielen. Von besonderem Interesse ist die Entdeckung von Tandem Repeats, also der mehrfachen Wiederholung eines bestimmten Sequenzmotivs. Tandem Repeats spielen eine zentrale Rolle bei einigen Krankheiten (Huntington Syndrom, Myogene Dystrophie, Friedrichs Atxia) und sind wahrscheinlich auch als Proteinbindungsstellen an der Genregulation beteiligt [7]. Tandemwiederholungen sind oft durch Mutationen in ihrer Struktur verändert. Von Rivals et al. [49] wurde ein Kompressionsverfahren vorgestellt, daß beurteilen kann, ob eine solche inexakte Tandemwiederholung signifikant ist oder nicht. Dabei wird eine Tandemwiederholung durch das Tupel (Motiv, Anzahl der Wiederholungen) sowie einer Liste von Mutationen repräsentiert. Wenn diese Darstellung kürzer ist, als die Länge des Motivs multipliziert mit der Anzahl der Wiederholungen, bezeichnet man diese (inexakte) Tandemwiederholung als signifikant.

Die Bestimmung der evolutionären Ähnlichkeit zweier Sequenzen ist ein schwieriges Problem in der Biologie. Es existieren zwar Algorithmen für das Sequenzalignment, aber diese können nur auf Sequenzen mit ähnlicher biologischer Funktion, wie verwandten Genen, angewendet werden. Eine weitere Einschränkung des Alignments ist die Unfähigkeit, mit evolutionären Ereignissen wie Duplikationen oder umgekehrten Komplementen umzugehen. Kompressionsverfahren umgehen diese Einschränkungen und können daher benutzt werden, um mit Hilfe bedingter Kompression ein Ähnlichkeitsmaß zu definieren. Die berechneten Distanzen können für die Rekonstruktion von phylogenetischen Bäumen benutzt werden. Dieses Thema wird in Kapitel 7 ausführlich behandelt.

Kapitel 5

Warum Standardkompressoren nicht funktionieren

In diesem Kapitel wird die Funktionsweise von verlustfreien Standardkompressionsalgorithmen erklärt, die in vielen Bereichen erfolgreich eingesetzt werden. Danach werden die Unterschiede von DNA zu menschlicher Sprache erörtert und aufgezeigt, warum es mit den vorgestellten Methoden nicht gelingt, DNA zu komprimieren. Am Ende des Kapitels werden die Ergebnisse dieser Verfahren verglichen. Für ausführlichere Erläuterungen der vorgestellten Methoden wird auf das Buch von Nelson verwiesen [42].

5.1 Wörterbuchmethoden

Wörterbuchmethoden basieren auf der Ersetzung von Zeichenketten durch ein einzelnes Kodewort. Dieses Kodewort ist entweder ein Zeiger auf dieselbe Zeichenkette an einer früheren Stelle im Text oder ein Pointer in ein Wörterbuch. Kompression wird erreicht, wenn dieses Kodewort kürzer als die ursprüngliche Darstellung ist.

5.1.1 Lempel Ziv 1977 (LZ77)

Dieses von Jacob Ziv und Abraham Lempel 1977 vorgestellte Verfahren [64] basiert auf einem adaptiven Wörterbuch, das als *Search Buffer* bezeichnet wird. Der Search Buffer besteht aus den k vorangehenden Zeichen der Positionen $p - k$ bis $p - 1$, wobei p die aktuellen Position im Text ist. Weiterhin existiert ein *Look*

Ahead Buffer, der die nächsten m Zeichen beginnend an der aktuellen Position im Text beinhaltet. LZ77 findet das längste Match eines Präfixes des Look Ahead Buffers mit einem Substring aus dem Search Buffer. In jedem Kodierschritt wird ein Tripel ausgegeben, welches aus Position und Länge des Matches sowie dem Zeichen, das dem Präfix folgt, besteht. Wenn kein Match mit einem Präfix gefunden wurde, speichert man als Länge eine 0. Um eine Position innerhalb n möglicher Positionen zu speichern, braucht man ein Kodewort mit mindestens $log_2(n)$ Bits. Deshalb sind für die Länge des Matches $log_2(m)$ Bits nötig, für die Position $log_2(k)$ Bits. Danach werden beide Puffer um die Anzahl der kodierten Zeichen verschoben.

Ein Problem bei LZ77 ist, daß auch lange und häufige Matches immer aus dem Puffer herausgeschoben werden und daher für weitere Vorkommen nicht verwendet werden können. Weiterhin ist die Länge eines Matches durch die Größe des Look Ahead Buffers beschränkt. Das Unix *gzip* basiert auf LZ77.

Bsp.: Der gegebene Text sei “aacabcaccabacaccb”, die aktuelle Position ist 9, k sei 7 und m sei 6 (“a|acabcac|cabaca|ccb”). Das längste Präfixmatch ist jetzt “cab” an Position 2 im Search Buffer. Als Tupel wird (2,3,“a”) ausgegeben und die nächste Position im Text ist 13.

5.1.2 Lempel Ziv 1978 (LZ78)

Ein Jahr später stellten Ziv und Lempel eine verbesserte Methode von LZ77 vor [65], die die angesprochenen Probleme behebt. LZ78 verwendet keinen Look Ahead oder Search Buffer. Man startet mit einem leeren Wörterbuch, das nur den leeren String enthält. Wie bei LZ77 versucht man den längsten Präfix zu finden, der im Wörterbuch vorkommt und kodiert ein Tupel, bestehend aus Pointer zu dem entsprechenden Wörterbucheintrag und dem danach folgenden Zeichen im Text. Der Wörterbucheintrag mit dem folgenden Zeichen wird jetzt als eine Zeichenkette ins Wörterbuch eingefügt. Durch diese Methode kann ein Match beliebig lang werden. Lange Redundanzen gelangen allerdings nur dann ins Wörterbuch, wenn diese häufig vorkommen, da die neu aufgenommene Zeichenkette immer nur um ein Zeichen länger ist, als der bisherige Wörterbucheintrag. Falls kein Match im Wörterbuch gefunden wurde, wird der leere String mit dem folgenden Zeichen kodiert. Ein String, der einmal eingefügt wurde, steht auch für spätere Verwendung zur Verfügung.

Bsp.: Der Inhalt des Wörterbuches nach der Verarbeitung des Textes aus dem vorherigen Beispiel sowie die ausgegebenen Tupel sind in Tabelle 5.1 gezeigt.

Wörterbuchindex	Wörterbucheintrag	ausgegebenes Tupel
0	null	
1	a	(0,a)
2	ac	(1,c)
3	ab	(1,b)
4	c	(0,c)
5	acc	(2,c)
6	aba	(3,a)
7	ca	(4,a)
8	cc	(4,c)
9	b	(0,b)

Tabelle 5.1: Wörterbuch nach Verarbeiten von “aacabcaccabacaccb”

5.1.3 Lempel Ziv Welch (LZW)

LZW [60] wurde 1984 von Welch als eine Verbesserung von LZ78 vorgeschlagen. Zu Beginn ist das Wörterbuch bei LZW nicht leer, sondern enthält die 256 ASCII–Zeichen. Dadurch ist es möglich, nur den Pointer zu dem Wörterbucheintrag ohne das folgende Zeichen zu kodieren. In das Wörterbuch wird danach der aktuelle Wörterbucheintrag zusammen mit dem nächsten Zeichen eingefügt.

Wenn ein Pointer aus 12 Bit besteht, beziehen sich also die Kodes 0 – 255 auf Einzelzeichen, die Kodes 256 – 4095 auf Zeichenketten. Eine weitere Verbesserung sind Pointer variabler Länge. Wenn das Wörterbuch nur 257 – 511 Einträge hat, braucht man nur 9 Bits, um einen Pointer zu speichern. Erst wenn mehr Einträge vorhanden sind, werden mehr Bits für einen Pointer benötigt. Eine Variante von LZW ist im Unix *compress* implementiert.

5.2 Statistische Methoden

Statistische Methoden basieren auf der Vorhersage von Wahrscheinlichkeiten für das folgende Symbol. Jedem Zeichen ordnet man einen eindeutigen Kode zu, wobei sehr wahrscheinliche Symbole kurze Kodewörter und wenig wahrscheinliche lange Kodewörter erhalten. Wenn die Vorhersage gut ist, erreicht man dadurch eine kleinere mittlere Kodewortlänge.

5.2.1 Arithmetische Kodierung

Arithmetische Kodierung [14, 40] benutzt Unterschiede in der Häufigkeit einzelner Symbole im Text, um Kompression zu erreichen. So kommt z.B. “e” im englischen Text viel öfter vor als “z”. Arithmetische Kodierung repräsentiert dabei den ganzen Text durch ein Intervall der reellen Zahlen zwischen 0 und 1.

Zu Beginn der Kodierung teilt man das halboffene Intervall der reellen Zahlen $[0, 1)$ in nicht überlappende Teilintervalle entsprechend der Wahrscheinlichkeit der Zeichen auf, d.h. große Wahrscheinlichkeiten ergeben große Teilintervalle. Jetzt wählt man für das nächste Symbol im Text das Teilintervall, welches diesem Zeichen zugeordnet ist. Dieses Teilintervall wird wieder in nicht überlappende Teilbereiche entsprechend der Zeichenhäufigkeiten aufgeteilt und der Vorgang solange wiederholt, bis alle Zeichen abgearbeitet worden. Das Ergebnis ist ein Zielintervall, das durch eine Zahl, die in diesem Intervall liegt, eindeutig identifiziert werden kann. Die Zahl mit der kürzesten Binärdarstellung aus dem Zielintervall, wird als Kodewort für den gesamten Text ausgegeben. Je größer das Zielintervall ist, desto weniger Bits sind für die Zahl nötig. Der Dekodierer kann anhand dieser Zahl mit derselben Verfahrensweise eindeutig das Zielintervall bestimmen und dadurch den Text rekonstruieren.

Bsp.: Ein Beispiel soll das Prinzip der arithmetischen Kodierung illustrieren. Das Alphabet besteht aus zwei Zeichen “a” und “b” mit Wahrscheinlichkeit $p(a) = 0.7$ und $p(b) = 0.3$. Es wird die Zeichenkette “aba” kodiert. Abbildung 5.1 zeigt die Schritte des Kodierers. Das Zielintervall ist $[0.49, 0.637)$. Ein möglicher Kode für diese Zeichenkette ist 0.1, was binär für 0.5 steht.

In der Praxis wird als letztes Zeichen ein sogenanntes *End of Message* Symbol kodiert, das dem Dekodierer signalisiert, daß der gesamte Text dekodiert worden ist. Weiterhin verwenden konkrete Implementierungen nicht das Intervall $[0, 1)$ der reellen Zahlen, sondern arbeiten beispielsweise auf dem Intervall $[0, 65536)$ der ganzen Zahlen. Intervallexpansion [42] verhindert, daß das Zielintervall nach vielen Schritten zu klein wird.

Arithmetische Kodierung ist nicht nur auf Wahrscheinlichkeiten einzelner Zeichen beschränkt, sondern zur Vorhersage für das folgende Zeichen kann auch ein Kontext beliebiger Länge verwendet werden (arithmetische Kodierung höherer Ordnung). Die verwendeten Wahrscheinlichkeiten sind entweder bekannt (*statische arithmetische Kodierung*) oder werden kontinuierlich aus den verarbeiteten Daten geschätzt (*adaptiv*). Man kann zeigen, daß arithmetische Kodierung für lange Zeichenketten eine mittlere Kodewortlänge erzeugt, die sich beliebig gut der Entropie nähert.

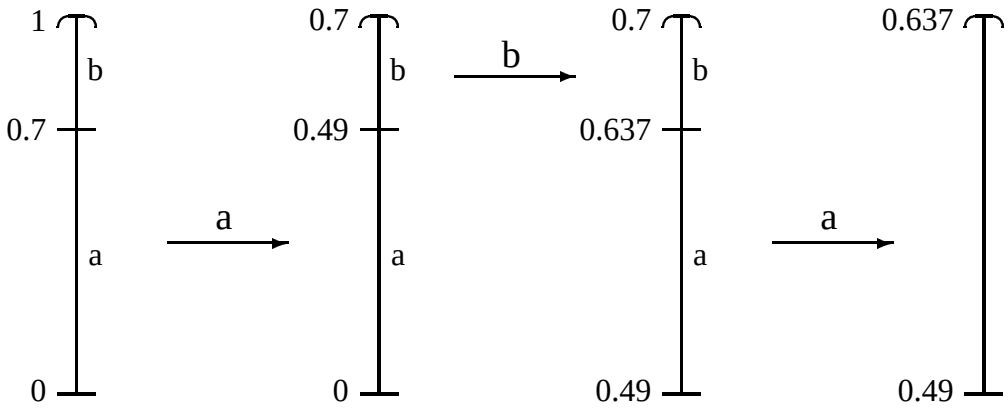


Abbildung 5.1: Prinzip der arithmetischen Kodierung

5.2.2 Prediction with Partial Matching (PPM)

PPM [15] ist ein statistischer Kompressor, der die Wahrscheinlichkeit des folgenden Zeichen aus einer Anzahl von vorhergehenden Zeichen (Kontext) schätzt. Wenn die Wahrscheinlichkeit der Textsymbole von den unmittelbar vorangehenden Zeichen abhängt (Markov Eigenschaft), erreicht man mit diesem Kompressor sehr hohe Kompressionsraten.

Die maximale Kontextlänge sei d. Für den aktuellen Kontext der Länge d bestimmt man die absolute Häufigkeit des Auftretens jedes Zeichens nach diesem Kontext anhand des bisher kodierten Textes. Aus dieser Statistik berechnet man die Wahrscheinlichkeit des folgenden Zeichens, was als Input für einen arithmetischen Kodierer dient.

Allerdings gibt es ein Problem zu beachten. Was passiert, wenn das zu kodierende Symbol nach diesem Kontext noch nie auftrat? Dann wäre die Wahrscheinlichkeit dieses Zeichens 0, und der arithmetische Kodierer würde ein unendlich langes Kodewort ausgeben. Zur Lösung wird ein spezielles *Escape* – Symbol *esc* kodiert, welches dem Dekodierer signalisiert, daß das zu kodierende Zeichen nach dem aktuellen Kontext noch nie vorkam. Der Kodierer wechselt daraufhin zum Kontext der Länge $d - 1$. Falls das Zeichen nach diesem Kontext auftrat, wird es mit der Wahrscheinlichkeit kodiert, die aus dem $d - 1$ langen Kontext berechnet wird. Ansonsten wird wieder *esc* kodiert. Der Vorgang wird gegebenenfalls solange wiederholt, bis der Kodierer beim leeren Kontext angelangt ist. Unter dem

leeren Kontext haben alle Symbole die gleiche Wahrscheinlichkeit. PPM ist eines der besten Kompressionsverfahren für englischen Text.

5.3 Unterschiede von DNA zu menschlicher Sprache

Viele der eben vorgestellten Kompressionsverfahren benutzen charakteristische Eigenschaften der menschlichen Sprache. Im englischen Text gibt es eine ungleiche Verteilung der einzelnen Buchstaben und Buchstabengruppen. Darüber hinaus kann man anhand eines Kontextes das folgende Zeichen sehr gut vorhersagen. Diese Markov Eigenschaft ist wesentlich für das Funktionieren von statistischen Kodierern, die mit Kontexten arbeiten (PPM). Redundanzen kommen in der Regel lokal vor und Wiederholungen sind oftmals exakt, wodurch Wörterbuchverfahren eine gute Kompression erreichen.

DNA Sequenzen weisen andere Charakteristika auf. Die Wahrscheinlichkeitsverteilung einzelner Nukleotide und Nukleotidgruppen ist oftmals sehr homogen. Redundanzen in der DNA sind oft nicht lokal, d.h. es gibt Wiederholungen, die an weit entfernten Stellen im Genom zu finden sind. Diese Eigenschaft verhindert eine Kompression mit Methoden, die auf festen Kontextlängen oder Wörterbüchern mit fixer Länge basieren (LZ77). Im Gegensatz zum englischen Text können diese Wiederholungen aber sehr lang sein und mehrere Tausend Basenpaare umfassen, was dem Kopieren von einigen Buchseiten gleich käme. Lange Wiederholungen kommen aber selten mehrmals vor, so daß auch LZ78 daraus keinen Nutzen ziehen kann, da lange Redundanzen nur ins Wörterbuch gelangen, wenn sie mehrfach vorkommen. Als DNA spezifische Besonderheit existieren umgekehrte, komplementäre Wiederholungen, die von keinem der bisher vorgestellten Verfahren berücksichtigt werden. Beispielsweise kommt in dem Chloroplastengenom der Tabakpflanze CHNTXX aus Tabelle 5.2 ein exaktes, umgekehrtes Komplement von 25341 Basenpaaren (bp) vor.

Eine weitere Schwierigkeit besteht darin, daß Mutationen im Laufe der Zeit exakte Repeats verändern, wodurch ursprünglich lange Wiederholungen nur noch als kurze, exakte Stücke erkannt werden können. Oftmals ist es dann sinnvoll, die gesamte inexakte Wiederholung zusammen mit einer Liste der Veränderungen zu kodieren.

Aufgrund der Degeneriertheit des genetischen Kodes, ist es nicht möglich, bei kodierenden Sequenzen nur die entstehenden Aminosäuren zu kodieren. Falls das machbar wäre, bräuchte man nur $log_2(20) = 4.32$ Bits um 3 Basen zu speichern und würde allein dadurch eine Kompressionsrate von $1 - (4.32/6) = 28\%$ er-

zielen. Wie in Abschnitt 2.2 beschrieben, kann man nicht von der Aminosäure auf das Triplet schließen, und wie in Kapitel 8 gezeigt wird, haben kodierende Abschnitte besonders hohe Entropien.

5.4 Ergebnisse

Wörterbuchverfahren

Die Ergebnisse von Wörterbuchverfahren bei der Kompression von DNA Sequenzen sind in Tabelle 5.2 dargestellt. Die dort gezeigten Sequenzen werden als inoffizielle Benchmarksequenzen verwendet und stellen die Vergleichbarkeit aller Kompressoren sicher. Als man sich etwa 1993 zum ersten Mal mit DNA Kompression beschäftigte, war die Menge der zur Verfügung stehenden Sequenzen noch sehr eingeschränkt. Diese Sequenzen wurden damals ausgewählt, weil sie ein einigermaßen breites Spektrum an Lebewesen abdeckten und hinreichend lang waren.

Bei den Sequenzen handelt es sich um ein sehr langes, menschliches Intron (HUMDYSTROP), 4 menschliche Gensequenzen (HUMGHCSA, HUMHBB, HUMHPRTB, HUMRETBLAS) sowie eine nichtkodierende, menschliche Sequenz (HUMHDABCD), das dritte Chromosom der Hefe (SCCHRIII), 2 komplette Mitochondriengenome (MPOMTCG, PANMTPACGA), 2 Chloroplastengenome (CHNTXX, CHMPXX) und 3 vollständige Virengenome (EBV, VACCG, HEHCMVCG). Die Länge dieser Sequenzen wird in Anzahl der Basenpaare (bp) angegeben. Alle Sequenzen sind unter dieser Bezeichnung in der Nukleotiddatenbank *GenBank* [17] verfügbar.

In Tabelle 5.2 werden die Entropien (in Bits pro Zeichen) von *gzip* und *compress* gezeigt. *Compress* steht dabei stellvertretend für eine LZW Implementation, *gzip* für LZ77. Gzip -9 korrespondiert zu *gzip* mit Option -9, wodurch man eine maximale Kompressionsrate bei *gzip* erreicht. Beide Programme sind Bestandteil einer Linux Standardinstallation.

Wenn DNA komplett zufällig wäre, würde ein optimaler Kode $log_2(4) = 2$ Bits brauchen, eine Base zu speichern. Bei allen Sequenzen benötigt *compress* mehr als 2 Bits pro Zeichen. Auch *gzip* expandiert das Outputfile in allen Fällen, abgesehen von EBV, was daran liegt, daß diese Sequenz viele lange Tandemwiederholungen hat. Diese Tandemwiederholungen sind lokale Redundanzen und können von *gzip* gewinnbringend gespeichert werden. Mit Option -9 erreicht *gzip* bessere Ergebnisse und kann auch HUMGHCSA komprimieren, eine Sequenz,

Sequenz	Größe	gzip	gzip -9	compress
Mensch				
HUMDYSTROP	38770	2.3633	2.3802	2.2333
HUMGHCSA	66495	2.0654	**1.5532**	2.1936
HUMHBB	73308	2.2454	2.2301	2.2001
HUMHDABCD	58864	2.2397	2.2112	2.2070
HUMHPRTB	56737	2.2669	2.2341	2.2024
HUMRETBLAS	180388	2.2724	2.2089	2.1436
Hefe				
SCCHRIII	315388	2.3297	2.2657	2.1821
Mitochondrien				
MPOMTCG	186609	2.3290	2.2807	2.2021
PANMTPACGA	100314	2.2925	2.2334	2.1164
Chloroplasten				
CHNTXX	155844	2.3347	2.2918	2.1870
CHMPXX	121024	2.2821	2.2212	2.0907
Viren				
EBV	172281	**1.8654**	**1.8508**	2.1653
VACCG	191737	2.2519	2.1904	2.1359
HEHCMVCG	229354	2.3277	2.2804	2.1990
Durchschnitt		2.2476	2.1737	2.1756

Tabelle 5.2: Vergleich von Wörterbuchverfahren

die eine außergewöhnlich hohe Zahl an Wiederholungen besitzt. Bei den restlichen 12 Sequenzen sind aber immer noch mehr als 2 Bits pro Zeichen notwendig.

Statistische Kompressionsverfahren

Die Ergebnisse von statistischen Kompressoren (Tabelle 5.3) sind etwas besser. Unter Verwendung von Funktionen der *QccPack Library* [20] habe ich vier Varianten eines arithmetischen Koders implementiert. *Arith* entspricht adaptiver arithmetischer Kodierung, die nur Häufigkeitsunterschiede von einzelnen Basen berücksichtigt. Arithmetische Kodierung erster, zweiter und vierter Ordnung sind als *arith1*, *arith2* und *arith4* bezeichnet.

Weiterhin wird eine PPM Implementierung betrachtet, die im Internet unter [30] zu finden ist. Die Kompressionsrate von PPM hängt von der Wahl der maximalen Kontextlänge d ab. Da es für jede Sequenz eine optimale Kontextlänge gibt,

Sequenz	Size	arith	arith1	arith2	arith4	PPM	CTW
Mensch							
HUMDYSTROP	38770	1.9480	1.9183	1.9190	1.9759	1.9369	1.9200
HUMGHCSA	66495	2.0002	1.9381	1.9338	1.9467	1.3028	1.3638
HUMHBB	73308	1.9656	1.9223	1.9174	1.9486	1.9258	1.8928
HUMHDABCD	58864	1.9950	1.9568	1.9433	1.9539	1.9398	1.8973
HUMHPRTB	56737	1.9704	1.9361	1.9265	1.9472	1.9386	1.9132
HUMRETBLAS	180388	1.9477	1.9141	1.9076	1.9174	1.9088	–
Hefe							
SCCHRIII	315388	1.9602	1.9525	1.9499	1.9569	1.9574	1.9450
Mitochondrien							
MPOMTCG	186609	1.9820	1.9679	1.9642	1.9771	1.9771	1.9624
PANMTPACGA	100314	1.8794	1.8709	1.8698	1.8877	1.8885	1.8664
Chloroplasten							
CHNTXX	155844	1.9545	1.9368	1.9344	1.9488	1.9415	1.9330
CHMPXX	121024	1.8601	1.8446	1.8430	1.8577	1.8403	1.8381
Viren							
EBV	172281	1.9661	1.9413	1.9386	1.9433	1.6897	–
VACCG	191737	1.9183	1.9149	1.9096	1.9185	1.9037	1.8577
HEHCMVCG	229354	1.9812	1.9753	1.9692	1.9685	1.9647	1.9584
Durchschnitt		1.9512	1.9279	1.9233	1.9406	1.8764	1.8623

Tabelle 5.3: Vergleich von statistischen Kompressionsverfahren

wurde d von 2 bis 16 variiert. Die in Tabelle 5.3 angegebenen Werte entsprechen der Kompressionsrate für die optimale Kontextlänge, die für alle Sequenzen 2 ist, mit Ausnahme von HUMGHCSA (13), EBV (15) und VACCG (3).

In der letzten Spalte werden die Ergebnisse für *Context Tree Weighting* (CTW) [61, 50] angegeben. CTW ist ein weiterer statistischer Kompressor, der ebenfalls zur Wahrscheinlichkeitsbestimmung Kontexte verwendet und sehr gute Kompressionsraten liefert. Im Gegensatz zu PPM verwendet man nicht nur einen Kontext, sondern bildet die Vorhersage, indem man die Wahrscheinlichkeiten des folgenden Symbols nach den Kontexten der Länge 1 bis d gewichtet aufsummiert, wobei d die maximale Kontextlänge ist. Diese Wahrscheinlichkeiten werden wieder arithmetisch kodiert. Die Ergebnisse sind dem Paper von Matsumoto, Sadakane und Imai [37] entnommen.

Im Gegensatz zu den Wörterbuchmethoden können alle statistischen Kom-

pressionsverfahren DNA Sequenzen komprimieren. Da die 4 Basen alle etwa gleichhäufig auftreten, erreicht *arith* nur eine sehr geringe Kompression, mit Ausnahme von CHMPXX und PANMTPACGA, in denen A und T viel häufiger sind (etwa 70 %) als C und G. Wenn man höhere Ordnungen betrachtet, werden die Ergebnisse besser, wobei Ordnung 2 die besten Resultate liefert. PPM ist in 8 von 14 Fällen schlechter als *arith2*, erreicht aber bei HUMGHCSA und EBV eine wesentlich höhere Kompression als die arithmetische Kodierung. CTW liefert die bisher besten Ergebnisse. Der in Tabelle 5.3 gezeigte Durchschnitt wurde nur aus den Werten berechnet, die für alle Methoden bekannt sind.

Wie Kapitel 6 zeigen wird, kann man mit einem Kompressor, der speziell für die DNA entwickelt wurde, eine deutlich höhere Kompressionsrate erzielen. Diese Algorithmen suchen neben Wiederholungen auch nach umgekehrten Komplementen und lassen inexakte Matches zu. Weiterhin werden auch weit entfernte und sehr lange Wiederholungen berücksichtigt. Außerdem überprüfen die meisten Verfahren vorher, ob es sich lohnt, eine Wiederholung abzuspeichern, oder ob der entstehende Kode länger als die Wiederholung selbst wäre. Durch diese Eigenschaft wird vermieden, daß die Dateigröße expandiert, wie das bei konventionellen Wörterbuchmethoden der Fall ist.

Kapitel 6

DNA Kompressionsverfahren

Seit etwa 1993 haben sich viele Forscher mit dem Problem der DNA Kompression und Entropieschätzung beschäftigt [1, 2, 4, 12, 21, 23, 24, 31, 33, 34, 35, 36, 37, 38, 39, 45, 46, 47, 48, 49, 53]. Einige der entstandenen Algorithmen sollen jetzt genauer vorgestellt werden. Dabei wird zwischen Entropieschätzern und echten Kompressionsverfahren unterschieden. Anhand der Standardsequenzen wird die geschätzte Entropie der Methoden verglichen und - so weit bekannt - Angaben über Laufzeitverhalten und Ressourcenanforderungen gemacht.

6.1 Entropieschätzer

Im Gegensatz zu einem Kompressor wird bei Entropieschätzern der letzte Schritt des Abspeicherns des entstehenden Kodes ausgelassen. Stattdessen schätzt man, wieviel Bits das Kodewort benötigen sollte. Ein weiterer Unterschied kann darin bestehen, daß nicht jedes Zeichen des Textes kodiert wird. In diesem Fall wählt man aus dem ganzen Text zufällig eine Anzahl von Zeichen und leitet aus deren Kompressionsrate die Entropie der gesamten Sequenz ab.

6.1.1 CDNA

Das 1997 von Loewenstern und Yianilos [36] vorgestellte Verfahren, basiert auf der Wahrscheinlichkeitsvorhersage des folgenden Zeichens anhand des aktuellen Kontextes. *CDNA* beschränkt sich dabei nicht nur auf die exakte Übereinstimmung eines Kontextes, sondern berücksichtigt auch die Wahrscheinlichkeitsverteilung der Zeichen nach ähnlichen Kontexten.

Dahinter steckt folgende Überlegung: Kontexte, die sich nur geringfügig unterscheiden, können Teile einer inexakten Wiederholung sein und sollten daher eine ähnliche Vorhersage für die folgende Base liefern. Weiterhin ermöglicht die Aufhebung der exakten Übereinstimmung die Verwendung von langen Kontexten. Das ist wünschenswert, da die Vorhersage des nächsten Zeichens wesentlich aussagekräftiger sein sollte, als dies bei kurzen Kontexten der Fall ist. Wenn man Kontexte der Länge 10 betrachtet, so gibt es bei der DNA Alphabetgröße 4, $4^{10} = 1.048.576$ verschiedene Kontexte. Um die Wahrscheinlichkeitsverteilung der Zeichen nach einem Kontext der Länge 10 gut zu schätzen, braucht man sehr lange Sequenzen, da alle 1.048.576 Kontexte genügend oft vorkommen müssen. Wenn man aber auch die Verteilung von ähnlichen Kontexten berücksichtigt, kann man schon aus kleinen Datenmengen vernünftige Statistiken ableiten. *CDNA* ist dadurch in der Lage, Kontexte bis zu einer Länge von 24 zu verwenden. Außerdem werden Statistiken von exakten und ähnlichen umgekehrten, komplementären Kontexten berücksichtigt.

CDNA Modell

CDNA berücksichtigt Kontexte verschiedener Länge und verschiedener Ähnlichkeit. Die Menge der erlaubten Kontextlängen wird als W bezeichnet. Zur Beurteilung der Ähnlichkeit zweier Kontexte wird der *Hammingabstand* herangezogen, der die Anzahl der Mismatches angibt; d.h. für einen gegebenen Kontext der Länge d, kann es andere Kontexte mit Hammingabständen 0 bis d geben.

Sei $match(S, T, h)$ die Menge der Matches des Strings S mit einem Teilstring gleicher Länge aus der Trainingssequenz T, wobei ein Match gerade den Hammingabstand h haben muß. Analog dazu sei $match(S, b, T, h)$ die Menge aller Substrings von T, die den Hammingabstand h zu S haben und nach denen das Zeichen b kommt. Der Suffix der Länge w von S wird als $suffix(S, w)$ bezeichnet. Für eine gegebene Sequenz T und einen Kontext S gibt es eine Wahrscheinlichkeitsverteilung $p_{w,h}(b|S, T)$, die angibt, mit welcher Wahrscheinlichkeit das Zeichen $b \in \{A, C, G, T\}$ nach einem Hammingabstand-h-Match von $suffix(S, w)$ in T vorkommt. Dieser Wahrscheinlichkeitsverteilung entspricht ein *Experte*, der S und T als Input bekommt und für ein bestimmtes w und h die Verteilung $p_{w,h}$ berechnet. Die Definition von $p_{w,h}$ lautet:

$$p_{w,h}(b|S,T) = \frac{1 + |match(suffix(S,w), b, T, h)|}{\sum_{\sigma \in \{A,C,G,T\}} 1 + |match(suffix(S,w), \sigma, T, h)|}$$

wobei $|.|$ die Anzahl der Elemente in einer Menge ist. Durch das Addieren der Eins wird erreicht, daß $p_{w,h}(b|S,T) > 0$, auch wenn $|match(suffix(S,w),b,T,h)| = 0$.

Bsp. 1: Gegeben sei T = "cagttacagcgatataagccattagaaacgc" sowie S = "aag". Damit ist $match(suffix(S,3),a,T,1) = \{tag\}$, $match(suffix(S,3),c,T,1) = \{cag,aaa,acg\}$, $match(suffix(S,3),g,T,1) = \{aac\}$ und $match(suffix(S,3),t,T,1) = \{cag\}$, womit sich $p_{3,1}(a|S,T) = p_{3,1}(g|S,T) = p_{3,1}(t|S,T) = 2/10$ und $p_{3,1}(c|S,T) = 4/10$ ergibt.

Bei der Vorhersage des folgenden Zeichens verwendet man nicht nur einen Experten, sondern berücksichtigt die Vorhersagen aller Experten $p_{w,h}$ für $w \in W$ und $h = 0,\ldots,w$. Die Gesamtvorhersage bildet man, indem die Einzelmeinungen gewichtet aufsummiert werden. Wenn die Summe der Gewichte 1 ist, erhält man wieder eine Wahrscheinlichkeitsverteilung, die ein arithmetischer Kodierer verwenden kann, um das folgende Zeichen zu kodieren.

Allerdings gibt es Fälle in denen $match(suffix(S,w),T,h)$ für ein gegebenes w die leere Menge ist. Dann würde $p_{w,h}(b|S,T)$ für alle $b \in \{A,C,G,T\}$ den Wert 0.25 liefern, was die beste Wahl ist, wenn keine Information zur Verfügung steht. Diese Experten sind damit *nicht informativ* und sollten das Gewicht 0 bekommen, um die Gesamtvorhersage nicht zu beeinflussen. Die Funktion $fh(w,S,T)$ gibt für eine Kontextlänge w den kleinsten Hammingabstand h zurück, für den $|match(suffix(S,w),T,h)| > 0$ gilt. Wenn T wenigstens so lang ist wie w, gibt es immer ein h für das $|match(suffix(S,w),T,h)| > 0$. Damit ist $fh(w,S,T)$ für alle w definiert. Bei einem hinreichend langen T ist es unwahrscheinlich, daß $|match(suffix(S,w),T,h)| = 0$ für ein $h > fh(w,S,T)$. Allerdings ist es häufig der Fall, daß bei einem langen Kontext kleine Hammingabstände kein Match liefern. Deshalb werden alle Experten $p_{w,fh(w,S,T)}, \ldots, p_{w,w}$ als informativ betrachtet und nur diese Experten bekommen Gewichte, die größer als 0 sind. Als Abkürzung für $fh(w,S,T)$ wird im Folgenden $fh(w)$ geschrieben.

Bsp. 2: Sei T dieselbe Sequenz aus dem vorherigen Beispiel und S="cccc". Dann ist $|match(suffix(S,4),T,0)| = |match(suffix(S,4),T,1)| = 0$ und $|match(suffix(S,4),T,2)| = |\{cagc,agcc,gcca,ccat,acgc\}| = 5$. Damit ist $fh(4) = 2$.

Die Vorhersage für das folgende Zeichen von S ist definiert als

$$P(b|S,T,\Upsilon,\Psi) = \sum_{w \in W} \sum_{h=fh(w)}^{w} p_{w,h}(b|S,T) \cdot \upsilon_{w,fh(w),h} \cdot \psi_w$$

wobei Υ und Ψ je ein Satz positiver Gewichte sind, deren Elemente $\sum_{h=fh(w)}^{w} \upsilon_{w,fh(w),h} = 1 \quad \forall\, w \in W$ und $\sum_{w \in W} \psi_w = 1$ erfüllen. Ψ sind die Gewichte für eine bestimmte Kontextlänge, Υ die Gewichte für alle informativen Experten.

Bsp. 3: Sei $W = \{2\}$ und $\Upsilon = ((0.7, 0.2, 0.1), (0.55, 0.45), (1))$. Wenn $fh(2) = 0$, dann erhält der Experte, der keine Mismatches zuläßt, ein sehr hohes Gewicht, wenn $fh(2) = 1$ sollen die Experten $p_{2,1}$ und $p_{2,2}$ etwa gleichgewichtet werden. Für den aktuellen Kontext S sei $fh(2) = 1$, $p_{2,1}(b|S,T) = 0.3$ und $p_{2,2}(b|S,T) = 0.35$. Bei Berücksichtigung des nicht informativen Experten, ist $P(b|S,T,\Upsilon,\Psi) = 0.7 \cdot 0.25 + 0.2 \cdot 0.3 + 0.1 \cdot 0.35 = 0.27$. Wenn man den nicht informativen Experten ignoriert, erhält man eine höhere Wahrscheinlichkeit $P(b|S,T,\Upsilon,\Psi) = 0.55 \cdot 0.3 + 0.45 \cdot 0.35 = 0.3225$ für das folgende Zeichen.

Die Wahrscheinlichkeit einer Sequenz U gegeben eine Trainingssequenz T sowie Gewichte Υ und Ψ ist definiert als

$$P(U|T,\Upsilon,\Psi) = \prod_{i=1}^{|U|} P(U[i] \,|\, U[1,i-1], T, \Upsilon, \Psi)$$

wobei $U[1, i-1]$ der Präfix von U bis zur Position $i-1$ ist. Für das aktuelle Zeichen an Position i bildet man also eine Vorhersage, indem man den aktuellen Kontext aus der bisher verarbeiteten Sequenz, die Trainingssequenz und die Gewichte verwendet.

Umgekehrte Komplemente

Bisher wurden nur Kontexte desselben DNA Stranges betrachtet. Bei DNA Sequenzen ist es aber sinnvoll, auch umgekehrte komplementäre Kontexte zu berücksichtigen, da der aktuelle Kontext Teil eines (inexakten) umgekehrten Komplements sein kann, das in der Sequenz vorkommt. Dann sollte vor einem Match des umgekehrten komplementären Kontextes das komplementäre Zeichen zu finden sein, das dem aktuellen Kontext folgt.

Die Funktion $rcmatch(suffix(S,w), b, T, h)$ liefert alle Substrings von T, die den Hammingabstand h zu dem umgekehrten, komplementären Suffix der

Länge w von S haben und vor denen das Komplement von b steht. Die neue Definition von $p_{w,h}(b|S,T)$ ist jetzt

$$p_{w,h}(b|S,T) = \frac{1 + Num(b)}{\sum_{\sigma \in \{A,C,G,T\}} 1 + Num(\sigma)}$$

wobei

$$Num(b) = |match(suffix(S,w), b, T, h)| + |rcmatch(suffix(S,w), b, T, h)|.$$

Bsp. 4: Die Sequenz T sei "acacgt g ccga taggt g ccga ggccg g ccga atggaa", der Kontext S="tcgg", und das auf S folgende Zeichen sei 'c'. Der String "tcggc" kommt dreimal als exaktes, umgekehrtes Komplement in T vor. $|rcmatch(suffix(S,4), c, T, 0)|$ ist damit 3 und $p_{4,0}(c|S,T) = 4/7$, da $Num(c) = 3$ und $Num(a) = Num(g) = Num(t) = 0$.

Trainingsphase

Um die Wahrscheinlichkeit $P(U|T, \Upsilon, \Psi)$ zu maximieren, sollte man den Experten hohe Gewichte zuordnen, die im Durchschnitt eine gute Vorhersage treffen. Da die Qualität der Experten sehr stark von den konkreten Sequenzen U und T abhängt, werden die Gewichte in jeder *Trainingsphase* neu bestimmt.

Dafür wählt man zufällig eine Position k in der Trainingssequenz T und extrahiert an dieser Stelle den Kontext $S = T[k - w_{max}, k - 1]$ (*Trainingssample*), wobei w_{max} die maximale Kontextlänge ist. Dann bestimmt man $p_{w,h}(T[k] \mid S, T')$ für alle w und h, wobei $T' = T[1 \ldots k-1, k+1 \ldots |T|]$ die gesamte Trainingssequenz bis auf Position k ist, da das Zeichen, das vorhergesagt werden soll, natürlich nicht bei der Bestimmung von $p_{w,h}(T[k] \mid S, T')$ berücksichtigt werden darf.

Dies wiederholt man für eine gegebene Anzahl von Trainingssamples. Sei K die Menge aller Positionen k, an denen eine Vorhersage gemacht wird. Ein Experte, der im Durchschnitt gute Vorhersagen trifft, liefert im Durchschnitt einen hohen Wert für $p_{w,h}(T[k] \mid S, T')$.

Zur Bestimmung der Gewichte wird der *Expectation Maximization (EM) Algorithmus* [18] verwendet. Der EM Algorithmus ist ein iterativer Optimierungsalgorithmus, der ein lokales Maximum findet und die vorteilhafte Eigenschaft hat, in jeder Iteration diesem Maximum ein Stück näher zu kommen, d.h. man gelangt

zielgerichtet zu einem lokalen Optimum der Gewichte. Zu Beginn sind alle Gewichte gleichverteilt. In jeder Iteration wird der Beitrag der einzelnen Experten an der Gesamtvorhersage (E – Step) berechnet und dann die Gewichte neu adjustiert (M – Step). Experten mit guter Vorhersage liefern einen großen Beitrag und werden in der nächsten Iteration entsprechend höher gewichtet. Für die Bestimmung der Gewichte reichen etwa 200 Iterationen bis zur Konvergenz.

Der EM Algorithmus besteht in jeder Iteration aus zwei Schritten. Zuerst bestimmt man den Beitrag jedes Experten

$$E\upsilon_{w,fh(w),h} = \sum_{k \in K} \frac{p_{w,h}(T[k] \mid S, T') \cdot \upsilon_{w,fh(w),h} \cdot \psi_w}{P(T[k] \mid S, T', \Upsilon, \Psi)}$$

an der Gesamtvorhersage. Dann berechnet man den Anteil aller Experten einer Kontextlänge w an der Gesamtwahrscheinlichkeit

$$E\psi_w = \sum_{h=fh(w)}^{w} E\upsilon_{w,fh(w),h} = \sum_{k \in K} \left[\sum_{h=fh(w)}^{w} \frac{p_{w,h}(T[k] \mid S, T') \cdot \upsilon_{w,fh(w),h} \cdot \psi_w}{P(T[k] \mid S, T', \Upsilon, \Psi)} \right]$$

wofür man die Anteile der einzelnen Experten für diese Kontextlänge aufsummiert. Im zweiten Schritt bestimmt man neue Gewichte Υ' und Ψ', wobei jeder Experte genau das Gewicht bekommt, das seinem Anteil an der Vorhersage entspricht. Die neuen Gewichte berechnet man durch

$$\upsilon'_{w,fh(w),h} = \begin{cases} \sum_{i=fh(w)}^{w} E\upsilon_{w,fh(w),i} = 0 & : \quad \frac{1}{w+1-fh(w)} \\ sonst & : \quad \frac{E\upsilon_{w,fh(w),h}}{\sum_{i=fh(w)}^{w} E\upsilon_{w,fh(w),i}} \end{cases}$$

sowie

$$\psi'_w = \frac{E\psi_w}{\sum_{i \in W} E\psi_i}$$

Nach einer Iteration gilt

$$\sum_{k \in K} P(T[k] \mid S, T', \Upsilon', \Psi') \geq \sum_{k \in K} P(T[k] \mid S, T', \Upsilon, \Psi)$$

Bei Gleichheit ist ein lokales Maximum erreicht und man bricht den Algorithmus ab. Andernfalls ersetzt man die alten Gewichte durch die neuen ($\Upsilon = \Upsilon'$ und $\Psi = \Psi'$) und beginnt die nächste Iteration.

Testphase

Das trainierte Modell, das aus den Gewichten der Experten besteht, wird in der *Testphase* benutzt, um die *Testsequenz* zu komprimieren. Die Wahrscheinlichkeit der Testsequenz U gegeben eine Trainingssequenz T ist $P(U|T, \Upsilon, \Psi)$. Man benötigt $-log_2(P(U|T, \Upsilon, \Psi))$ Bits, um U zu kodieren. Die geschätzte Entropie (pro Zeichen) ist dann $-log_2(P(U|T, \Upsilon, \Psi))/|U|$. Das Ablaufschema von *CDNA* ist in Abbildung 6.1 dargestellt.

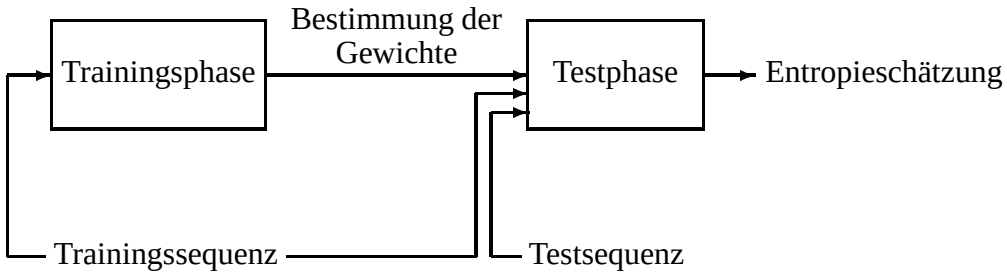


Abbildung 6.1: Schema von CDNA

Allerdings ist es sehr rechenaufwendig und zeitintensiv, jedes Zeichen der Sequenz U zu kodieren. Deshalb kodiert man nur eine Anzahl von zufällig gewählten Zeichen aus U und schätzt aus deren durchschnittlicher Kodewortlänge die Entropie der gesamten Sequenz.

Dazu extrahiert man an der zufälligen Position k in U den Kontext $U[k - w_{max}, k - 1]$ (*Testsample*) und berechnet $p_{w,h}(U[k] \mid U[k - w_{max}, k - 1], T)$ für alle w und h, wobei T jetzt wieder die vollständige Trainingssequenz ist. Die Anzahl an Bits, die gebraucht werden, um $U[k]$ zu kodieren, ist

$$-log_2(P(U[k] \mid U[k - w_{max}, k - 1], T, \Upsilon, \Psi))$$

Damit ergibt sich die Entropieschätzung für U aus der aufsummierten Anzahl der Bits für alle Testsamples geteilt durch deren Anzahl

$$H(U|T) = \frac{\sum_{k \in K} -log_2(P(U[k] \mid U[k - w_{max}, k - 1], T, \Upsilon, \Psi))}{|K|}$$

wobei K die Menge der Positionen ist, an denen Zeichen der Testsequenz kodiert werden. Bei ausreichend hoher Samplezahl liefert diese Approximation eine

genaue Schätzung und ist trotzdem wesentlich schneller, als die vollständige Kodierung von U.

Praktische Anwendung

Bisher wurde noch nicht erklärt, aus welchem Teilen der Eingabesequenz die Trainings– und Testsequenz besteht. Im einfachsten Fall stammen Trainings– und Testsequenz von verschiedenen Regionen im Genom oder von unterschiedlichen Lebewesen. Mit dieser Cross–Entropieschätzung kann man auf den Grad der Verwandschaft der beiden Sequenzen Rückschlüsse ziehen. Wenn die Testsequenz sehr unterschiedlich zur Trainingssequenz ist, dann liefern die Experten schlechte Vorhersagen, was zu einer hohen Entropie führt. Bei einer niedrigen Entropie kann man auf gemeinsame Eigenschaften der Sequenzen schließen.

Normalerweise möchte man aber die Entropie einer Sequenz ermitteln, ohne eine Trainingssequenz vorzugeben. In diesem Fall müssen Trainings– und Testsequenz Teile der Inputsequenz sein. Wie das genau geschieht, wird jetzt erläutert.

Statistische Kompressionsverfahren brauchen oft lange Sequenzen, um eine gute Entropieschätzung zu liefern, weil am Anfang der Sequenz noch nicht ausreichend viele, statistische Informationen über den Input vorliegen. Daher überschätzt ein echter Kompressor die Entropie fast immer. Um diesen Nachteil auszugleichen, könnte man statistische Informationen aus dem Anfang der Sequenz sammeln und diese dann auf den übrigen Text anwenden. Für die Entropieschätzung würde man dann nur die Kompressionsrate des Textendes berücksichtigen. Hierbei läuft man Gefahr, daß sich das Ende wesentlich vom restlichen Text unterscheidet und die Schätzung nicht für den gesamten Text gültig wäre.

Loewenstern und Yianilos schlagen deshalb *Crossvalidierung* vor. Dabei teilt man die Sequenz in n gleiche Teile, trainiert das Modell mit beliebigen $n - 1$ Teilen, komprimiert dann das fehlende Stück und schätzt damit die Entropie. Dies wird für alle n Teile wiederholt, und der Durchschnitt bildet die Entropieschätzung der ganzen Sequenz. Allerdings ist es bei diesem Ansatz möglich, die Entropie zu unterschätzen. Man stelle sich eine Sequenz vor, die aus zwei gleichen, aber zufälligen Teilen besteht. Die statistischen Eigenschaften beider Teile sind damit gleich. Crossvalidierung mit 2 Segmenten wird jetzt eine Entropie nahe Null liefern, aber die wahre Entropie ist etwa ein Bit pro Zeichen. Auch wenn dieses Beispiel konstruiert wirkt, gibt es doch eine nicht geringe Wahrscheinlichkeit für das Auftreten einer solchen Sequenz, wenn man an die vielen Tandem Repeats im menschlichen Genom denkt.

Um diesen Nachteil zu umgehen, kann man eine andere Vorgehensweise an-

wenden, die im Folgenden als *CDNA compress* bezeichnet wird. Dabei teilt man die Sequenz wieder in n Teile auf. Der erste Teil wird mit 2 Bits pro Zeichen kodiert, was dem kürzesten Kodewort entspricht, wenn keine Trainingsinformationen vorliegen. Mit den statistischen Informationen aus dem ersten Teil, wird der zweite Teil komprimiert. Danach trainiert man das Modell mit den ersten zwei Teilen und kodiert dann den dritten Teil, usw.. Als Schätzung wird der Durchschnitt der Entropien aller Teile genommen.

6.1.2 Grammar Transform Analysis and Compression (GTAC)

Dieser Entropieschätzer wurde im Jahr 2000 von Lanctot, Li und Yang [31] publiziert. Der Algorithmus benutzt *kontextfreie Grammatiken*, um die Entropie einer DNA Sequenz zu schätzen. Dabei wird die Eingabesequenz zuerst in eine geeignete, kontextfreie Grammatik transformiert, aus der dann eine Entropieschätzung abgeleitet wird.

Kontextfreie Grammatiken und Grammatiktransformationen

Kontextfreie Grammatiken (Chomsky Typ 2) bestehen aus einem Quadrupel $G = (V, T, R, S)$, wobei V die Menge der Variablen, T die Menge der Alphabetzeichen, R die Menge aller Regeln und S das Startsymbol ist. Die Regeln sind von der Form $u \rightarrow v$ und ersetzen eine Variable durch einen String, bestehend aus neuen Variablen und Alphabetzeichen. Für kontextfreie Grammatiken muß gelten: $u \in V$ und $v \in (V \cup T)^*$, wobei $(.)^*$ die reflexive, transitive Hülle einer Menge ist. Kontextfreiheit bedeutet, man kann eine Regel für eine Variable aus V anwenden, ohne den Kontext dieser Variable berücksichtigen zu müssen.

Aus einer Grammatik kann man eine Menge von Wörtern erzeugen, indem man sukzessiv Regeln anwendet, wobei man mit einer Regel beginnt, die das Startsymbol S auf der linken Seite hat. Als Sprache bezeichnet man die Menge aller Wörter, die mit einer Grammatik erzeugt werden können.

Bsp. 1: Sei $G = (\{S, T\}, \{a, b\}, \{S \rightarrow aST, S \rightarrow aT, T \rightarrow b\}, S)$. Die Ableitung eines Wortes könnte lauten $S \overset{S \rightarrow aST}{\longrightarrow} aST \overset{S \rightarrow aST}{\longrightarrow} aaSTT \overset{S \rightarrow aT}{\longrightarrow} aaaTTT \overset{T \rightarrow b}{\longrightarrow} aaabTT \overset{T \rightarrow b}{\longrightarrow} aaabTb \overset{T \rightarrow b}{\longrightarrow} aaabbb$, was das Wort “aaabbb” ergibt. Die erzeugte Sprache von G ist $\{ab, aabb, aaabbb, \ldots\}$.

Durch eine *Grammatik–Transformation* $x \rightarrow G_x$ wird eine Eingabesequenz x in eine kontextfreie Grammatik G_x umgewandelt. Diese Grammatik muß dabei die folgende Bedingungen erfüllen:

- aus G_x ist nur x ableitbar
- G_x ist deterministisch, d.h. jede Variable erscheint nur einmal auf der linken Seite aller Regeln
- die Regeln aus R enthalten keinen leeren String auf der rechten Seite
- G_x hat keine unnützen Regeln, d.h. bei der Erzeugung von x wird jede Regel mindestens einmal benutzt.

Da die erzeugte Sprache von G_x nur aus x besteht, ist die gesamte Information von x in der Grammatik G_x enthalten.

Bsp. 2: Die Grammatik $G_x = (\{S, A, B, C, D\}, \{a, c, g, t\}, R, S)$ mit Regelmenge $R = \{S \to ACAD, A \to acBC, B \to gg, C \to ata, D \to Bgt\}$ erzeugt eindeutig den String x = "acggataataacggatagggt".

Entropieschätzung anhand kontextfreier Grammatiken

Man kann eine Grammatik G_x arithmetisch kodieren und dann die Entropie von x schätzen. Wie dies genau geschieht, soll hier nicht besprochen werden. Stattdessen soll die Kompressionsrate von Grammatiken theoretisch untersucht werden, um daraus einen guten Entropieschätzer zu konstruieren.

Dazu wird G_x zunächst auf den String $w(G_x)$ reduziert, der gebraucht wird, um die *unnormalisierte Entropie* $H(G_x)$ von G_x zu berechnen. An die rechte Seite der Regel, die mit dem Startsymbol S beginnt, hängt man die rechte Seite einer anderen Regel an. Aus dem resultierenden String löscht man das erste Vorkommen der Variablen auf der linken Seite der eben angehängten Regel. Anschließend wiederholt man dies für die restlichen Regeln. Der String $w(G_x)$ sei nun die rechte Seite der mit S beginnenden Regel.

Bsp. 3: Für die oben angegebene Grammatik ergibt sich nach Anhängen der rechten Seite der ersten Regel

$$S \to ACADacBC,$$

nach Löschen der Variablen A

$$S \to CADacBC$$

und nach Verarbeiten der zweiten Regel

$$S \to CADacCgg.$$

Wenn alle 4 Regeln verarbeitet sind, ist $w(G_x)$ = "AacCggataBgt".

Nach folgender Formel wird die *unnormalisierte Entropie* von $H(G_x)$ berechnet.

$$H(G_x) = \sum_{s \in w(G_x)} n(s)\, log_2 \frac{|w(G_x)|}{n(s)}$$

wobei $n(s)$ die Anzahl des Vorkommens des Symbols s in $w(G_x)$ ist und $|w(G_x)|$ die Länge von $w(G_x)$.

Bsp. 4: Für das obige Beispiel gilt $n(A) = n(B) = n(C) = n(c) = 1$, $n(t) = 2$ und $n(a) = n(g) = 3$. Damit ergibt sich $H(G_x) = 4 \cdot log_2(12) + 2 \cdot log_2(6) + 2 \cdot 3 \cdot log_2(4) = 31.51$.

Das folgende Theorem (Theorem 3.1 in [31]) zeigt, wie die unnormalisierte Entropie einer Grammatik zur Schätzung der Entropie von x benutzt werden kann.

Theorem 6.1 *Für jede Grammatik G_x existiert ein eindeutig dekodierbares Kodewort $B(G_x)$ mit*

$$|B(G_x)| = H(G_x) + f(G_x)$$

wobei $|B(G_x)|$ die Länge des Kodewortes und $f(G_x)$ ein Overhead–Term ist.

Als Schätzer für die Entropie kann damit $H(G_x)/|x|$ verwendet werden, also die unnormalisierte Entropie geteilt durch die Länge von x. Um korrekt zu sein, müßte man noch einen Overhead–Term $f(G_x)$ berücksichtigen, der allerdings eine obere Schranke besitzt und verschwindet, wenn die Länge der Sequenz steigt. Dieser Overhead würde im Allgemeinen zu einer Überschätzung der Entropie führen und wird daher vernachlässigt.

Man sucht jetzt nach einer grammatischen Transformation, die für x eine Grammatik G_x findet, deren unnormalisierte Entropie $H(G_x)$ so klein wie möglich ist. Dafür sind *irreduzible Grammatik–Transformationen* interessant. Eine Grammatik ist irreduzibel, wenn

- jede Variable außer S mindestens zweimal auf den rechten Seiten aller Regeln vorkommt
- es keine Wiederholung mit zwei oder mehr Zeichen auf den rechten Seiten aller Regeln gibt
- und jede Variable einen unterschiedlichen Substring von x erzeugt.

Eine irreduzible grammatische Transformationen transformiert jeden Input x in eine irreduzible Grammatik.

Die Konvergenzrate dieses Entropieschätzers ist schwer zu bestimmen und hängt von den Eigenschaften der Quelle sowie der verwendeten grammatischen Transformation ab. Daher versuchen Lanctot, Li und Yang nicht die Konvergenzrate zu bestimmen, sondern verfolgen einen anderen Weg. Sie zeigen im folgenden Theorem (Theorem 3.3 in [31]), daß unabhängig von der irreduziblen grammatischen Transformation und den Eigenschaften der Quelle dieser Entropieschätzer die wahre Entropie fast nie unterschätzt.

Theorem 6.2 *Sei $X^n = X_1 \dots X_n$ die Ausgabe einer beliebigen Quelle. Dann gilt für eine Konstante $d > 0$ mit Wahrscheinlichkeit $p \geq 1 - n^{-d}$:*

$$\frac{H(G_{X^n})}{n} \geq -\frac{1}{n} log_2(P(X^n)) - \frac{f(G_{X^n})}{n} - \frac{d\, log_2(n)}{n}$$

für jede irreduzible grammatische Transformation $X^n \rightarrow G_{X^n}$, wobei $P(X^n)$ die Wahrscheinlichkeit von X^n ist.

Der Term $-log_2(P(X^n))/n$ ist dabei die Entropie von X^n und steht für die wahre Entropie der Quelle (was korrekt ist, wenn n sehr groß). $f(G_{X^n})/n$ ist klein und besitzt im schlimmsten Fall eine obere Schranke von $O(1/log_2(n))$. Das Theorem sagt also aus, daß mit hoher Wahrscheinlichkeit der Entropieschätzer die wirkliche Entropie nicht wesentlich unterschätzen wird.

Eine weitere Aussage des Theorems ist, daß der beste Schätzer durch die irreduzible Grammatik–Transformation gegeben ist, die den kleinsten Wert für $H(G_x)$ liefert. Allerdings ist das Finden dieser Transformation NP–hart und damit die optimale Grammatik–Transformation praktisch nicht berechenbar. Dafür können die Autoren zeigen, daß es möglich ist eine irreduzible grammatische Transformation in Linearzeit zu konstruieren. Dieser Algorithmus wird Grammar Transform Analysis and Compression (*GTAC*) genannt.

GTAC Algorithmus

Der *GTAC* Algorithmus basiert auf der wiederholten Lösung des *längsten, nicht überlappenden Pattern (LNP) Problems*. Dabei sucht man den längsten Substring einer Sequenz, der mindestens zweimal vorkommt und dessen Vorkommen sich nicht überlappen. So ist in der Sequenz “agtagtagtagt” das LNP “agtagt”, während der längste, wiederholte Substring “agtagtagt” ist.

GTAC beginnt mit der trivialen Grammatik, die nur aus einer einzigen Regel

$$A \rightarrow x$$

besteht, wobei x die gesamte Eingabesequenz ist.

Sei β der längste, nicht überlappende Substring von x, der mindestens zweimal vorkommt. Dann kann man x in der Form $x = \alpha_1\ \beta\ \alpha_2\ \beta\ \alpha_3$ schreiben. *GTAC* erzeugt jetzt eine neue Regel für β und ersetzt β durch eine neue Variable.

$$A \rightarrow \alpha_1\ B\ \alpha_2\ B\ \alpha_3$$
$$B \rightarrow \beta$$

Das gleiche kann auch geschehen, wenn β Teil von zwei verschiedenen Regeln ist, wie z.B.

$$A \rightarrow \alpha_1\ \beta\ \alpha_2$$
$$B \rightarrow \alpha_3\ \beta\ \alpha_4$$

was ebenfalls eine neue Regel für β ergibt

$$A \rightarrow \alpha_1\ C\ \alpha_2$$
$$B \rightarrow \alpha_3\ C\ \alpha_4$$
$$C \rightarrow \beta$$

Weiterhin löst *GTAC* das LNP Problem auch für umgekehrte Komplemente und ersetzt entweder eine normale Wiederholung oder ein umgekehrtes Komplement, je nachdem was länger ist. Wenn keine neue Regel mehr erstellt werden kann, hat man eine irreduzible grammatische Transformation für x gefunden und kann aus deren unnormalisierter Entropie die wahre Entropie schätzen.

6.1.3 Match Length Entropieschätzer

Es gibt mehrere Wege, die Entropie einer Sequenz zu schätzen. Zum einen kann man die Definition der Entropie $\hat{H} = -\sum_i \hat{p}_i\, log_2(\hat{p}_i)$ zur Schätzung verwenden, was eine hinreichend genaue Schätzung der Wahrscheinlichkeiten $\hat{p}_i$ für alle Zeichen voraussetzt. Die Genauigkeit der Entropieschätzung hängt dabei allein von der Genauigkeit der Wahrscheinlichkeiten ab. Im Allgemeinen braucht man sehr lange Sequenzen, um eine gute Schätzung der Zeichenwahrscheinlichkeiten zu bekommen.

Weiterhin kann man die durchschnittliche Kodewortlänge eines Kompressors als obere Schranke für die Entropie ansehen. Aber auch die bisher vorgestellten Kompressionsverfahren und Entropieschätzer haben eine langsame Konvergenz. So sind selbst kurze englische Texte mit LZ78 nicht komprimierbar, da Kompression erst dann erreicht wird, wenn man ein geeignetes Wörterbuch aufgebaut hat.

Daher ist die Bestimmung der Entropie für kurze Sequenzen äußerst problematisch, weil in den meisten Fällen die wahre Entropie überschätzt wird. Es ist aber wünschenswert, auch für kürzere Sequenzen (wie z.B. Exons und Introns, siehe Kapitel 8) die Entropie genau bestimmen zu können. Von Farach et al. [21] (für ausführlichere Erläuterungen siehe [66]) wurde deshalb der Match Length Entropieschätzer entwickelt. Dieser Entropieschätzer konvergiert sehr schnell und liefert daher auch für kurze Sequenzen sinnvolle Schätzungen. Der Match Length Entropieschätzer basiert auf einem LZ77 ähnlichen Verfahren und verwendet Pattern Matching. Die Herleitung dieses Schätzers erfordert zunächst eine Erklärung der *Recurrence Time*.

Recurrence Time

Als X_1^l soll im Folgenden die Sequenz $X_1, X_2, \ldots, X_l$ bezeichnet werden.

Definition 6.1 (Recurrence Time) *Die Recurrence Time von X_1^l in einer Quelle wird als N_l bezeichnet und ist definiert als die kleinste, ganze Zahl N, so daß $X_1^l = X_{N+1}^{N+l}$.*

Das frequentistische Verständnis der Wahrscheinlichkeit eines Ereignisses definiert diese als Anzahl des Auftretens des Ereignisses geteilt durch die Summe aller Ereignisse. Dann ist die Zeit zwischen zwei Ereignissen (Recurrence Time) umgekehrt proportional zur Wahrscheinlichkeit des Ereignisses.

Sei ein beliebiger String S und eine Sequenz T gegeben. Die Quellen, die S und T erzeugen, seien stationär und ergodisch. Bei stationären Quellen sind die statistischen Eigenschaften unabhängig vom Zeitpunkt. Ist die Quelle zusätzlich noch ergodisch, kann man durch (eventuell langes) Beobachten der Ausgabe der Quelle auf die zugrundeliegende Wahrscheinlichkeitsverteilung schließen.

Das folgende Lemma [66] sagt aus, daß die Recurrence Time von S in T umgekehrt proportional zu $P(S)$, der Wahrscheinlichkeit von S, ist.

Lemma 6.2.1 (Kac's Lemma) *Die erwartete Zeit bis zum Wiederauftreten eines festen Patterns S in einer stationären, ergodischen Quelle ist $1/P(S)$.*

Bsp. 1: Die Quelle soll zufällig Zeichen aus dem Alphabet $\{a, b, c, \ldots, x, y, z\}$ ausgeben. Dann ist die Wahrscheinlichkeit des Patterns "entropie" gleich 26^{-8} (da das Pattern 8 Zeichen lang ist) und im Durchschnitt kommt dieser String aller 26^8 Zeichen vor. Die Entropie dieser Quelle ist $H = log_2(26)$. Damit kann man 26^{-8} als $2^{-8\,log_2(26)} = 2^{-8H}$ schreiben und die Recurrence Time als 2^{8H}.

Theorem 6.3 (Asymptotisches Äquipartitions Theorem) *Sei eine Quelle gegeben mit Wahrscheinlichkeitsverteilung P und eine von dieser Quelle generierte Sequenz X_1^n. Dann gilt für alle $\epsilon > 0$:*

$$P(|-\frac{1}{n}log_2(P(X_1^n)) - H(P)| \geq \epsilon) \stackrel{n\to\infty}{\longrightarrow} 0.$$

Dieses Theorem sagt aus, das für große n gilt: $-\frac{1}{n}log_2(P(X_1^n) \approx H$. Damit kann man die Recurrence Time für die Entropieschätzung verwenden, was das nächste Theorem [66] aussagt.

Theorem 6.4 (Recurrence Time Theorem) *Sei N_l die Zeit bis zum ersten Wiederauftreten von X_1^l in einer stationären, ergodischen Quelle. Dann gilt*

$$\lim_{l\to\infty} \frac{log_2 N_l}{l} = H$$

mit Wahrscheinlichkeit 1.

Die Recurrence Time eines genügend langen Patterns liefert also eine genaue Schätzung der Entropie. Allerdings benötigt man immer noch große Datenmengen, um ein langes Pattern zweimal zu finden. Es kann also passieren, daß $N_l > n$ für eine gegebene Sequenz X_1^n.

Match Length Entropieschätzer

Um einen schnell konvergierenden Entropieschätzer zu finden, muß man diesen Nachteil beseitigen. Dazu verlagert man die Bestimmung der Recurrence Time auf die Bestimmung des *längsten Matches eines Präfixes* der folgenden Sequenz mit den bisherigen Beobachtungen.

Definition 6.2 (Matchlänge) *Die Matchlänge L_n sei definiert als die Länge des längsten, exakten Matches des Präfixes von X_1^∞ mit den letzten n zurückliegenden Zeichen X_{-n+1}^0.*

Die Matchlänge ist äquivalent zur Recurrence Time, weil $\{N_l > n\} = \{L_n < l\}$. Das heißt, die Menge der n, für die es kein zweites Auftreten von X_1^l in X_1^n gibt ($\{N_l > n\}$), ist gleich der Menge der n, für die es kein Präfixmatch der Länge l mit den letzten n Beobachtungen gibt ($\{L_n < l\}$).

Bsp. 2: Sei X_1^n = "abbacaabaaccaabba" mit $n = 17$ und $l = 4$. Dann ist die Recurrence Time N_4 von X_1^4 = "abba" gleich 13, da $X_1^4 = X_{14}^{17}$. Für $n = 1, \ldots, 12$ ist $N_4 > n$. Für die Bestimmung der Matchlänge teilt man die Sequenz in die folgenden Zeichen X_1^∞ = "abba..." und die vorangegangenen $n = 13$ Zeichen X_{-12}^0 = "abbacaabaacca" auf. Dann ist $L_{13} = 4$, da "abba" das längste Match ist. Für $n = 1, \ldots, 12$ ist $L_n < 4$.

Zu dem Recurrence Time Theorem gibt es ein Äquivalent, das zeigt, wie die Matchlänge zur Entropiebestimmung verwendet werden kann.

Theorem 6.5 (Match Length Theorem) *Sei L_n die Länge des längsten, exakten Matches des Präfixes von X_1^∞ mit den letzten n Beobachtungen X_{-n+1}^0, Dann konvergiert*

$$\lim_{n\to\infty} \frac{L_n}{log_2 n} = \frac{1}{H}$$

in Wahrscheinlichkeit.

In [66] wird gezeigt, daß für Quellen mit endlichem Gedächtnis

$$\frac{EL_i(n)}{log_2 n} = \frac{1}{H} + \frac{O(1)}{log_2 n}$$

wobei $L_i(n)$ das längste Match von X_i^∞ mit den letzten n Zeichen X_{i-n}^{i-1} ist und E der Erwartungswert. Bei Quellen mit endlichem Gedächtnis hängt die Wahrscheinlichkeit des nächsten Zeichens nur von einer endlichen Anzahl der vorangegangenen Zeichen ab, formal $P(X_k = x_k | X_{-\infty}^{k-1} = x_{-\infty}^{k-1}) = P(X_k = x_k | X_{k-M}^{k-1} = x_{k-M}^{k-1})$. Dies gilt auch, wenn das Gedächtnis der Quelle (M) sehr groß ist.

Um die Entropie einer Sequenz zu schätzen, geht man folgendermaßen vor. Man berechnet $L_i(n)$ für alle Positionen, die einen Abstand von n Zeichen zu den Enden der Sequenz haben. Anschließend wird der Mittelwert $\overline{L}$ aller L_i berechnet.

Als Schätzer für die Entropie einer Sequenz verwendet man dann

$$\hat{H} = \frac{log_2 n}{\overline{L}}$$

Die Schätzung konvergiert dabei mit einem Fehler von $O(1/log_2 n)$ und hängt auch vom Standardfehler von $\overline{L}$ (Standardabweichung der Mittelwerte) ab. Je mehr zurückliegende Zeichen zur Verfügung stehen, desto genauer wird die Schätzung der Entropie. Der Grund für die schnelle Konvergenz des Schätzers liegt in der Berechnung der Matchlänge für jedes Zeichen des Textes, während

LZ77 nach der Kodierung eines Matches das Schiebefenster um die Länge des Matches verschiebt.

Bei Anwendung des Schätzers auf DNA Sequenzen wird dabei die Annahme gemacht, daß die DNA Quelle ein endliches Gedächtnis hat. Wie groß das Gedächtnis dieser Quelle wirklich ist, läßt sich nur schwer sagen. Auch gelten die bisherigen Betrachtungen nur für stationäre und ergodische Quellen. DNA Sequenzen stammen wahrscheinlich nicht aus einer stationären Quelle, da die Häufigkeit der Basen in verschiedenen Genomabschnitten unterschiedlich sein kann. Trotzdem kann dieser Entropieschätzer bei DNA Sequenzen erfolgreich verwendet werden, um Entropieunterschiede zwischen Exons und Introns aufzudecken, was in Kapitel 8 ausführlich behandelt wird.

Es sei bemerkt, daß dieser Entropieschätzer nicht speziell für DNA Sequenzen entwickelt wurde und daher auch keine DNA spezifischen Merkmale wie umgekehrte Komplemente oder inexakte Wiederholungen berücksichtigt. Aus diesen Gründen wird der Match Length Entropieschätzer auch nicht bezüglich der Entropie von Standardsequenzen in Kapitel 6.3 verglichen. Der Sinn dieses Schätzers liegt in der schnellen Konvergenz, deren Verhalten beweisbar ist und der Anwendung auf kurze Sequenzen.

Pseudocode

Nachfolgend habe ich den Pseudokode für den Match Length Entropieschätzer angegeben. Obwohl es Linearzeit–Algorithmen (z.B. Boyer–Moore) für Stringvergleiche gibt, habe ich für das Pattern Matching die triviale Brute Force Implementierung verwendet. Da n im Allgemeinen klein ist (z.B. $n = 16$) und der Schätzer wegen seiner schnellen Konvergenz auf kurze Sequenzen angewendet werden soll, ist diese Implementierung trotzdem sehr schnell.

```
// berechne Li(n) fuer Sequenz
int computeL(char Sequenz[], integer Position, integer n)
{
   integer i, j, k, match;
   integer L = 0;

   k = 0;
   for (j = Position - n ... Position - 1){
      match = 0;

      for (i = 0 ... n-k-1){
         if (Sequ[j+i] == Sequ[Position+i])
```

```
            match++;
         else
            break;
      }

      k++;
      if (match > L)
         L = match;
   }
   return L;
}


// berechne den Match Length Entropieschaetzer fuer Sequenz
void computeMLEntropyEstimator(char Sequenz[], integer n)
{
   integer i;
   double L = 0, averageL = 0;

   // berechne alle Li
   for (i = n ... |Sequenz|-n) {
      L = L + computeL(Sequenz, i, n);
   }

   averageL = L / (|Sequenz| - 2*n + 1);
   print "Entropie: " + log2(n)/averageL;
}
```

6.2 Kompressionsverfahren

Verlustfreie Kompressionsverfahren erzeugen ein Outputfile, das den Kode für die komplette Inputsequenz enthält. Mit einem entsprechenden Dekodierer ist es möglich, aus dieser Datei den exakten Originaltext wiederherzustellen. Aus der Größe des Outputs und Inputs kann man die Entropie schätzen. In den meisten Fällen wird dabei die Entropie überschätzt, dafür hat man aber die Sicherheit, daß die wahre Entropie nie größer als die Schätzung sein kann.

6.2.1 Biocompress II

Biocompress II war der erste Kompressor, der DNA wirklich komprimieren konnte und deutlich geringere Entropieschätzungen lieferte als arithmetische Kodie-

rung höherer Ordnung. Der 1994 von Grumbach und Tahi [24] entwickelte Algorithmus erkennt sowohl exakte Wiederholungen als auch exakte, umgekehrte Komplemente und verwendet arithmetische Kodierung.

Im Gegensatz zu LZ77 besteht der Search Buffer bei *Biocompress II* nicht nur aus einer festen Anzahl von Zeichen, sondern ist die gesamte, bisher kodierte Sequenz. Der Kodierer sucht an der aktuellen Position im Text nach dem längsten Match des Präfixes mit der bisher kodierten Sequenz. Ein Match kann entweder eine Wiederholung oder ein umgekehrtes Komplement sein.

Das längste Match wird dann durch ein Tripel (l, p, c) repräsentiert, wobei l die Länge, p die Position des ersten Vorkommens im Search Buffer und c ein Bit zur Unterscheidung von umgekehrten Komplementen und Wiederholungen ist. Um den Präfix der Länge l mit 2 Bits pro Base abzuspeichern (*Literalkodierung*), benötigt man $2\,l$ Bits. Sei m die Anzahl an Bits, die gebraucht werden, um das Tripel (l, p, c) zu speichern. Dieses Tripel ermöglicht Kompression, wenn $m + K < 2\,l$, wobei K die Länge eines *Kontrollkodewortes* ist, das benutzt wird, damit ein folgendes Tripel von der Literalkodierung unterschieden werden kann. Falls dies erfüllt ist, speichert man das Tripel und verschiebt die aktuelle Position um die Länge des Matches. Ob das Tripel gespeichert wird, hängt entscheidend von der Länge des Matches sowie der Position ab.

Da l und p einen sehr großen Wertebereich haben können (von 0 bis zur aktuellen Position), verwendet man keine Kodewörter mit fester Länge, sondern *Fibonacci Kodes* [24], um l und p zu speichern. Fibonacci Kodes sind die effizientesten, selbstbeschränkenden Kodes variabler Länge, d.h. im Kode ist eine eindeutige Kennung enthalten, die das Ende eines Kodewortes signalisiert. Diese eindeutige Kennung sind zwei aufeinanderfolgende Einsen, die nur am Ende des Kodewortes vorkommen dürfen. Die Fibonacci Kodewörter für die Zahlen 1 bis 8 sind in Tabelle 6.1 angegeben.

Falls ein Match nicht lang genug oder so weit von der aktuellen Position entfernt ist, daß die Speicherung des Tripels keinen Gewinn bringt, fügt *Biocompress II* das aktuelle Zeichen in einen Puffer ein und geht zur nächsten Position. Wenn anschließend wieder ein Match kodiert werden soll, wird vorher geprüft, ob der Puffer leer ist. Ist dies nicht der Fall, speichert man die im Puffer enthaltenen Zeichen entweder mit *adaptiver, arithmetischer Kodierung der Ordnung 2* oder als Literalkodierung, je nachdem was weniger Bits benötigt. Um zwischen den drei Kodierungsvarianten (Match, arithmetische Kodierung, Literalkodierung) unterscheiden zu können, wird das Kontrollkodewort K benutzt. Im Falle von arithmetischer oder Literalkodierung wird noch die Anzahl der ausgegebenen Bits bzw. die Anzahl der Zeichen abgespeichert.

Zahl	Fibonacci Kodewort	Kodewortlänge
1	11	2
2	011	3
3	0011	4
4	1011	4
5	00011	5
6	10011	5
7	01011	5
8	000011	6

Tabelle 6.1: Fibonacci Kodewörter

Der Programmablaufplan von *Biocompress II* wird in Abbildung 6.2 gezeigt. Mit $gain(l, p)$ habe ich die Funktion bezeichnet, die entscheidet, ob die Speicherung eines Tripels im Vergleich zur Literalkodierung Bits spart. Die Funktionen $arith(buffer)$ bzw. $literal(buffer)$ liefern die arithmetische bzw. Literalkodierung von dem Puffer; $|buffer|$ gibt die Anzahl der Zeichen im Puffer an.

Biocompress II hebt damit Beschränkungen von LZ77 oder LZ78 auf. Matches können beliebig weit weg und so lang wie der Search Buffer sein. Weiterhin werden umgekehrte Komplemente berücksichtigt. Zeichen, die nicht Teil eines Matches sind, werden arithmetisch kodiert, um Unterschiede in der Häufigkeit von Dinukleotiden auszunutzen. Weiterhin wird darauf geachtet, daß durch arithmetische Kodierung und der Speicherung von Matches die entstehende Datei nicht expandiert wird.

6.2.2 GenCompress

GenCompress [12] arbeitet nach einem ähnlichen Schema wie *Biocompress II*. Allerdings gibt es einen wesentlichen Unterschied. Der von Chen, Kwong und Li 1999 veröffentlichte Algorithmus läßt nicht nur exake Wiederholungen und umgekehrte Komplemente zu, sondern benutzt auch *inexakte Matches*, um eine höhere Kompression zu erreichen.

Wie bei *Biocompress II* besteht der Search Buffer bei *GenCompress* aus der bisher kodierten Sequenz. Sei $w = vu$ die Eingabesequenz, wobei v der Search Buffer und u die noch zu kodierende Sequenz ist. Um ein inexaktes Match eines Präfixes s von u mit einem Substring t aus v zu speichern, benötigt man nicht nur das Tripel (l, p, c) für Länge, Position, Bit für Wiederholung bzw. umgekehrtes

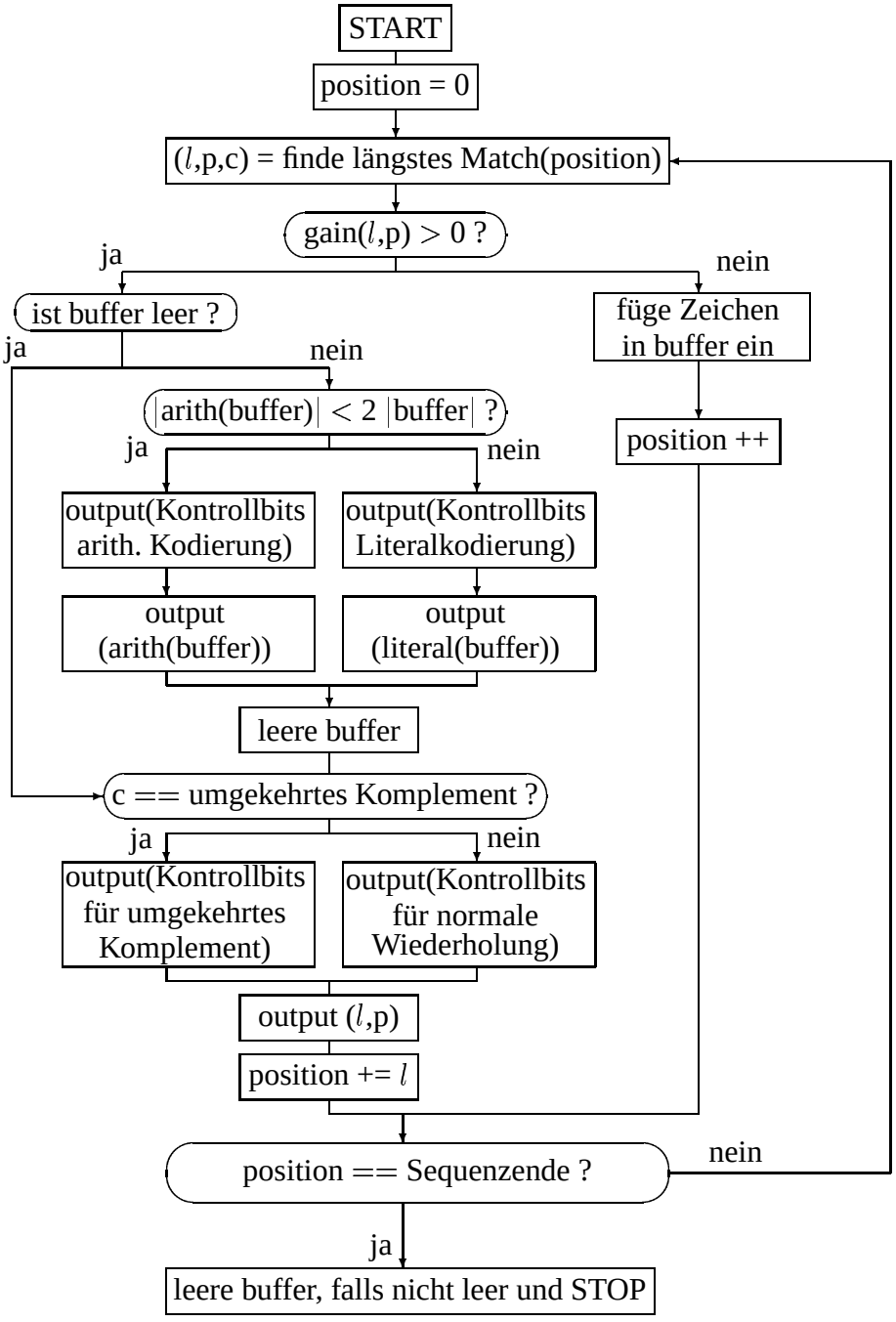


Abbildung 6.2: Programmablaufplan von Biocompress II

Komplement, sondern zusätzlich noch eine Liste von *Edit Operationen* $\lambda(s,t)$, die s in t überführen.

Mögliche Edit Operationen sind Ersetzungen (Mutation), dargestellt durch das Tripel $(R,p,char)$, Insertionen $(I,p,char)$ und Deletionen (D,p), wobei p die zu ändernde Position in s ist und $char$ das neue Zeichen für diese Position. $\lambda(s,t)$ besteht also aus einer Liste von Tripeln (Ersetzungen, Insertionen) und Tupeln (Deletionen). Die Gesamtanzahl an Bits, die benötigt werden, um ein inexaktes Match zu speichern, ist

$$m = |(l,p,c)| + |\lambda(s,t)| + K$$

wobei $|(l,p,c)|$ und $|\lambda(s,t)|$ die Bitzahl für das Match und die Edit Liste ist und K wieder die Bitzahl eines Kontrollkodewortes. Die Funktion $gain$ ist jetzt definiert als

$$gain(l,p,\lambda) = max\{2l - m, 0\}$$

und gibt an, ob durch Speicherung des inexakten Matches Bits gespart werden.

Der wesentliche Unterschied zwischen *Biocompress II* und *GenCompress* ist das Ausnutzen von inexakten Wiederholungen. Ein Beispiel (Abb. 6.3) zeigt, wie eine inexakte Wiederholung von *Biocompress II* nur als 3 kürzere, exakte Wiederholungen erkannt wird, während *GenCompress* die gesamte Wiederholung mit einer Liste von Edit Operationen kodiert. Die obere Teilsequenz unterscheidet sich von der unteren nur durch zwei Mutationen in Position 72,73 und einer Deletion an Position 90. Dabei wird unterstellt, das es keine “bessere” Wiederholung der unteren Sequenz im 2000 Zeichen langen Search Buffer gibt.

Ein optimaler (inexakter) Präfix s maximiert $gain(s,t,\lambda)$. Nun kann jeder Präfix s von u der Länge l in einen Substring t aus v der Länge j überführt werden, wofür man maximal $max\{l,j\}$ Edit Operationen braucht. Um diesen sehr großen Suchraum einzuschränken, suchen Chen, Kwong und Li nur nach Präfixen, die folgende Bedingung erfüllen. Ein Präfix s erfüllt die Bedingung $C = (k,b)$, wenn jeder Substring von s der Länge k nicht mehr als b Edit Operationen braucht. Damit wird sichergestellt, daß kurze Substrings in s nicht zu viele Edit Operationen verbrauchen, da es dann wahrscheinlich sinnvoller ist, s an dieser Stelle aufzuteilen und zwei getrennte Matches zu kodieren. In *GenCompress* ist $k = 12$ und $b = 3$. Es werden also höchstens 3 Edit Operationen für einen 12 Zeichen langen Substring zugelassen. Weiterhin beginnt und endet ein optimaler Präfix immer mit einem kurzen, exakten Teilmatch, da es nicht sinnvoll ist, Edit Operationen an den Enden zuzulassen. Es gibt mehrere $\lambda(s,t)$, die s in t überführen. Ein optimaler Präfix verwendet immer die Edit Liste, die die kleinste Anzahl an Bits benötigt.

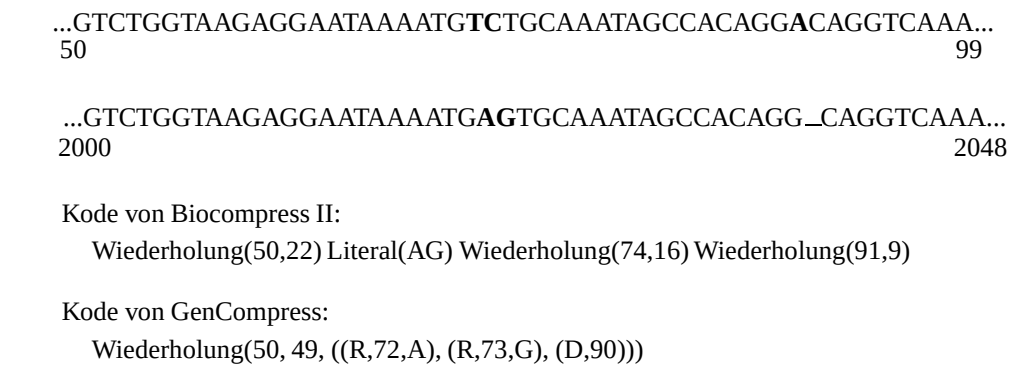


Abbildung 6.3: Kodierung von inexakten Wiederholungen bei Biocompress II und GenCompress

Bsp.: Sei s = "aact" und t = "aagt". Dann ist $\lambda = ((I, 3, g), (D, 3))$ eine gültige, aber nicht optimale Edit Liste. Die optimale Edit Liste $\lambda = ((R, 3, g))$ würde nur aus einer Operation bestehen.

Falls es keinen aktuellen Präfix gibt, für den $gain(s, t, \lambda) > 0$ gilt, fügt *GenCompress* das aktuelle Zeichen in einen Puffer ein und geht zur nächsten Position. Analog zu *Biocompress II* wird dieser Puffer mit *adaptiver, arithmetischer Kodierung zweiter Ordnung* geleert, falls anschließend wieder ein Match kodiert werden soll. *GenCompress* verwendet immer arithmetische Kodierung und prüft nicht vorher, ob Literalkodierung weniger Bits benötigt.

Der Programmablaufplan (Abb. 6.4) von *GenCompress I* unterscheidet sich nicht wesentlich von *Biocompress II* und ist etwas übersichtlicher, da die Pufferleerung ausschließlich mit arithmetischer Kodierung erfolgt. Allerdings ist die Implementation der optimalen Präfixsuche wesentlich schwieriger als ein exakter Stringvergleich in *Biocompress II*.

Chen, Kwong und Li implementierten zwei verschiedene Versionen. *GenCompress I* läßt als Edit Operationen nur Ersetzungen zu, d.h. es wird nach inexakten Matches gesucht, die einen möglichst geringen Hammingabstand zueinander haben. *GenCompress II* verwendet alle Edit Operationen (Ersetzungen, Insertionen und Deletionen). In praktischen Tests zeigte sich eine leichte Überlegenheit der ersten Version, allerdings ist der Unterschied äußerst gering.

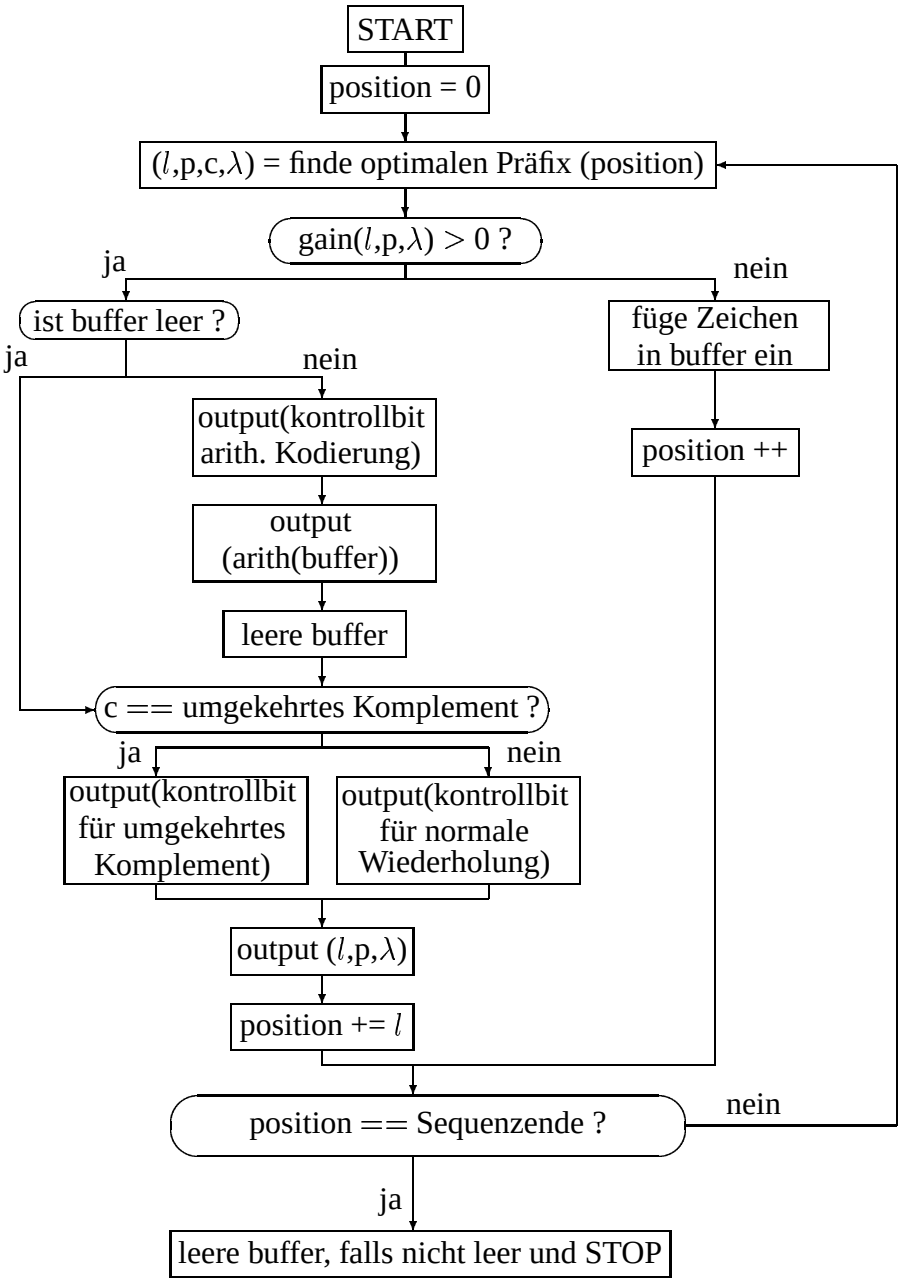


Abbildung 6.4: Programmablaufplan von GenCompress

6.3 Ergebnisse

DNA Entropieschätzer

In Tabelle 6.2 sind die Resultate der DNA Entropieschätzer dargestellt. Die verwendeten Sequenzen sind die gleichen wie in Kapitel 5.4.

CDNA crossval entspricht *CDNA* mit Crossvalidierung, *CDNA compress* der Variante, die am Ende von Kapitel 6.1.1 beschrieben wird. Zur Erstellung der Ergebnisse in den ersten beiden Spalten der Tabelle habe ich *CDNA* in C++ nachimplementiert. *CDNA* benötigt viele Parameter. In den Tests wurden folgende Werte verwendet:

- der EM Algorithmus wurde nach 200 Iterationen abgebrochen
- die erlaubten Kontextlängen waren 2,4,6,8, . . . ,24
- für *CDNA crossval* wurde eine 8–fache Crossvalidierung (8 gleiche Segmente für jede Eingabesequenz) durchgeführt mit je 1000 Samples pro Segment, d.h. es wurde mit 7000 Samples trainiert und die Entropieschätzung anhand von 1000 Testsamples erstellt
- bei *CDNA compress* wurde die Sequenz in 20 gleiche Segmente aufgeteilt und je 5000 Trainings– und Testsamples gezogen

Diese Parameter wurden auch von Loewenstern und Yianilos [36] verwendet. Die hohe Anzahl an Trainings– und Testsamples stellt sicher, daß die Schätzung repräsentativ für die ganze Sequenz ist. Bis auf geringe Abweichungen konnte ich mit meiner *CDNA* Implementation die Ergebnisse in [36] bestätigen.

Die letzte Spalte entspricht den Werten für *GTAC* aus [31].

Wie man es erwarten würde, ist *CDNA compress* in allen Fällen schlechter als *CDNA crossval*, da das erste Segment mit 2 Bits pro Zeichen kodiert wird. *GTAC* ist in 5 von 12 Fällen besser als *CDNA crossval* und in 10 von 12 Fällen besser als *CDNA compress*, wobei man aber die Gefahr der Entropieunterschätzung durch *CDNA crossval* gerade bei Sequenzen mit vielen Tandem Repeats (EBV) beachten muß. Im Durchschnitt (nur über die für alle Entropieschätzer vorhandenen Werte) erreicht *CDNA crossval* die besten Resultate. Die Ergebnisse sind um Größenordnungen besser, als die der statistischen Kompressoren oder Wörterbuchverfahren aus Kapitel 5.

CDNA ist sehr rechenaufwendig, da für jedes Sample die ganze Trainingssequenz durchlaufen werden muß. Für kurze Sequenzen wie HUMGHCSA oder

Sequenz	Size	CDNA crossval	CDNA compress	GTAC
Mensch				
HUMDYSTROP	38770	1.9066	1.9284	1.81
HUMGHCSA	66495	0.5054	0.9728	1.10
HUMHBB	73308	1.6746	1.7760	1.73
HUMHDABCD	58864	1.6476	1.6780	1.70
HUMHPRTB	56737	1.6726	1.7276	1.72
HUMRETBLAS	180388	1.6713	1.7435	–
Hefe				
SCCHRIII	315388	1.9075	1.9222	1.82
Mitochondrien				
MPOMTCG	186609	1.8265	1.8739	1.78
PANMTPACGA	100314	1.8374	1.8399	1.74
Chloroplasten				
CHNTXX	155844	1.3004	1.6275	1.53
CHMPXX	121024	1.5345	1.6849	1.58
Viren				
EBV	172281	1.5388	1.6101	–
VACCG	191737	1.6855	1.7990	1.67
HEHCMVCG	229354	1.7341	1.8675	1.74
Durchschnitt		1.6028	1.7248	1.66

Tabelle 6.2: Vergleich der DNA Entropieschätzer

HUMHDABCD braucht *CDNA crossval* 3 – 4 Stunden auf einem SGI Origin 200 Rechner mit MIPS R12000 270 MHz CPU und 1,5 GB RAM. Längere Sequenzen wie VACCG oder CHNTXX benötigen 9 – 11 Stunden, das 315388 bp große dritte Hefechromosom sogar 19 Stunden. Durch die höhere Anzahl an Samples und Segmenten braucht *CDNA compress* noch mehr Rechenzeit (kurze Sequenzen 4 – 5 Stunden, längere Sequenzen 15 – 18 Stunden, SCCHRIII 31 Stunden). Der Speicherbedarf von *CDNA* für diese Sequenzen und die verwendeten Parameter liegt bei etwa 40 – 55 MB Arbeitsspeicher und hängt von der Samplezahl sowie der Anzahl der Kontextlängen ab.

DNA Kompressionsverfahren

Da ein echter Kompressionsalgorithmus immer einen gewissen Overhead braucht, um die exakte Rekonstruktion der Sequenz zu ermöglichen, sollten die Bitraten

Sequenz	Size	Biocompress II	GenCompress	CTW+LZ
Mensch				
HUMDYSTROP	38770	1.9262	1.9231	1.9175
HUMGHCSA	66495	1.3074	1.0971	1.0972
HUMHBB	73308	1.8768	1.8134	1.8082
HUMHDABCD	58864	1.8770	1.8091	1.8218
HUMHPRTB	56737	1.9066	1.8299	1.8433
HUMRETBLAS	180388	–	1.7899	–
Hefe				
SCCHRIII	315388	1.9226	1.9168	–
Mitochondrien				
MPOMTCG	186609	1.9378	1.8980	1.9000
PANMTPACGA	100314	1.8752	1.8609	1.8555
Chloroplasten				
CHNTXX	155844	1.6172	1.6138	1.6129
CHMPXX	121024	1.6848	1.6697	1.6690
Viren				
EBV	172281	–	1.5394	–
VACCG	191737	1.7614	1.7614	1.7616
HEHCMVCG	229354	1.8480	1.8472	1.8414
Durchschnitt		1.7835	1.7385	1.7389

Tabelle 6.3: Vergleich der DNA Kompressionsverfahren

der Entropieschätzer immer niedriger sein. Tabelle 6.3 zeigt die Ergebnisse für *Biocompress II*, *GenCompress* und *CTW+LZ*. Die Ergebnisse für *Biocompress II* sind dem Paper von Grumbach und Tahi [24] entnommen. In der Spalte von *GenCompress* werden die Ergebnisse für *GenCompress II* gezeigt. Eine ausführbare Version von *GenCompress II*, die unter [28] heruntergeladen werden kann, wurde zur Kompression der Benchmarksequenzen verwendet.

CTW+LZ ist ein von Matsumoto et al. (2000) entwickelter Kompressor, der in [37] genauer beschrieben wird. Wie *GenCompress* verwendet *CTW+LZ* eine Kombination von einem statistischen Kompressor und dem Kodieren von inexakten Matches (Wiederholungen oder umgekehrten Komplementen). An Stelle von arithmetischer Kodierung kommt das in Kapitel 5.4 kurz erwähnte *Context Tree Weighting* (CTW) zum Einsatz. Der zweite Unterschied zu *GenCompress* ist die verwendete Lazy–Strategie. *CTW+LZ* speichert nicht sofort den optimalen Präfix,

sondern überprüft, ob an den 32 folgenden Positionen nicht ein (inexaktes) Match beginnt, das zu einem größeren Gewinn führt. Anschließend wird entweder das beste Match der 32 Positionen gespeichert oder CTW für die folgenden Zeichen verwendet. Die in Tabelle 6.3 gezeigten Ergebnisse stammen aus dem Paper von Matsumoto et al. [37].

Im Vergleich zu *Biocompress II* erreicht *GenCompress* eine deutlich höhere Kompressionsrate, was auf das Zulassen von inexakten Matches zurückzuführen ist. Das zeigt, daß inexakte Wiederholungen ein wesentlicher Bestandteil von DNA Sequenzen sind.

CTW+LZ ist in 6 von 11 Fällen besser als *GenCompress*, allerdings unterscheiden sich die Ergebnisse oft erst in der dritten Nachkommastelle. Im Durchschnitt, der wieder nur über die Werte gebildet wurde, die für alle Kompressoren vorhanden sind, gleichen sich aber beide Verfahren.

Vorteile von *CTW+LZ* sind zum einen die bessere Performance von CTW gegenüber arithmetischer Kodierung und die verwendete Lazy–Strategie, die der Greedy–Strategie von *GenCompress* überlegen sein dürfte. Allerdings gibt es große Unterschiede in der benötigten Rechenzeit. Für kurze Sequenzen (HUMDYSTROP) braucht *GenCompress* auf einem Rechner mit Intel Pentium III 450 MHz CPU und 256 MB RAM gerade 10 Sekunden, während Matsumoto angibt, auf einer SUN Workstation 8 Minuten zu benötigen. Für längere Sequenzen wird der Unterschied noch gravierender. *GenCompress* kann HEHCMVCG in 1 Minute und PANMTPACGA in 2 Minuten komprimieren. Für diese Sequenzen braucht *CTW+LZ* mehrere Stunden. Trotzdem steigt auch die Laufzeit von *GenCompress* exponentiell, wenn die Sequenzlänge steigt. In Anbetracht der vergleichbaren Kompressionsraten und der deutlichen Geschwindigkeitsvorteile ist *GenCompress* der zur Zeit beste DNA Kompressor.

Kapitel 7

Vergleich von Genomen

Kompressionsverfahren können zum Vergleich von Genomen verwendet werden. Anhand dieser Informationen kann man Aussagen über den evolutionären Abstand zweier Organismen machen, was zur Rekonstruktion von phylogenetischen Bäumen benutzt werden kann. In diesem Kapitel geht es zuerst um Distanzmatrizen und phylogenetische Bäume. Anschließend wird ein Algorithmus zur Erstellung von Stammbäumen erklärt. Danach werden bisherige Methoden zur Definition von Abstandsmaßen vorgestellt und gezeigt, wie man mit Hilfe der Kolmogorov Komplexität und der bedingten Kompression ein Distanzmaß für DNA Sequenzen definieren kann. Die Qualität dieses Abstandsmaßes wird an einem praktischen Beispiel getestet, woran sich eine Diskussion der Vor– und Nachteile anschließt.

7.1 Distanzmatrizen und Phylogenetische Bäume

Heutige Evolutionsmodelle gehen meist von einer baumartigen Evolution aus. Dabei sind zwei Arten aus einer einzigen Art hervorgegangen, indem sie sich getrennt weiterentwickelt haben. Austausch von Genen, wie dies bei Bakterien beobachtet werden kann oder Kreuzung zwischen den Arten wird hierbei nicht berücksichtigt. Aus dem Grad der Ähnlichkeit zweier Organismen kann man auf Verwandtschaftsverhältnisse Rückschlüsse ziehen sowie den *Zeitpunkt der Aufspaltung zweier Arten* schätzen. Dahinter steckt die Überlegung, daß zwei Arten um so größere Unterschiede aufweisen, je mehr Zeit seit dem Beginn ihrer getrennten Weiterentwicklung vergangen ist.

Ein *Phylogenetischer Baum* soll die *Phylogenie* (Stammesgeschichte) von Le-

bewesen anschaulich machen. Zur Rekonstruktion von phylogenetischen Bäumen gibt es 3 unterschiedliche Ansätze:

1. Distanzbasierte Methoden
2. Maximum Parsimony Methoden
3. Maximum Likelihood Methoden

Im Folgenden werde ich nur auf die Distanzbasierten Methoden eingehen. Für ausführlichere Erläuterungen und Beweise, der hier gemachten Aussagen, verweise ich auf [26].

Definition 7.1 (Distanzmatrix) *Eine Distanzmatrix D ist eine symmetrische n x n Matrix mit $D(i,i) = 0$ für alle $i = 1, \ldots, n$, wobei $D(i,j)$ den Abstand von Art i zu Art j angibt.*

Man unterscheidet 3 verschiedene Klassen von Distanzen: *ultrametrische, additive* und *nicht additive* Distanzen.

Ultrametrische Distanzen

Eine ultrametrische Matrix D erfüllt das sogenannte *Dreipunktkriterium*, das besagt:
Für alle $X_1, X_2, X_3 \in \{1, \ldots, n\}$ gibt es eine Nummerierung A,B,C, so daß

$$D(A,B) \leq D(A,C) = D(B,C).$$

Von 3 Arten ist die maximale Distanz also nicht eindeutig.

Definition 7.2 (ultrametrischer Baum) *Ein ultrametrischer Baum zur ultrametrischen n x n Distanzmatrix D erfüllt folgende Bedingungen:*

- *es gibt genau n Blätter und jeder innere Knoten hat mindestens 2 Kinder*
- *jeder innere Knoten ist mit einem Wert aus D beschriftet, und diese Werte sind auf einem Pfad von der Wurzel zu einem Blatt streng monoton fallend*
- *$D(i,j)$ ist der Wert des inneren Knotens, der den nächsten gemeinsamen Vorfahren von Art i und j darstellt.*

Ein ultrametrischer Baum ist immer gerichtet und alle Blätter haben den selben Abstand zur Wurzel. Zu jeder ultrametrischer Matrix D gibt es genau einen zugehörigen ultrametrischen Baum [26].

Ein Beispiel eines ultrametrischen Baumes ist in Abbildung 7.1 gezeigt. Dort ist der Knoten, der A und B verbindet, der gemeinsame Vorfahre von A, B. Die Wurzel ist der Vorfahre von allen drei Arten. Dabei entspricht v dem Eintrag $D(A, B)$, u dem Eintrag $D(A, C) = D(B, C)$, wobei $u > v$ gelten muß. Es ist leicht einzusehen, daß $D(A, B) \leq D(A, C) = D(B, C)$, womit das Dreipunktkriterium erfüllt ist.

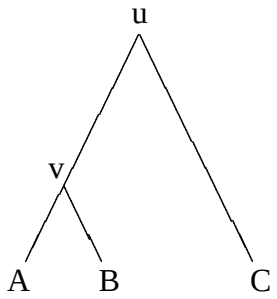


Abbildung 7.1: Ein einfacher ultrametrischer Baum

Ein ultrametrischer Baum hat folgenden Bezug zu einem phylogenetischen Baum. Jedes Blatt stellt eine heute existierende Art dar. Die inneren Knoten bezeichnen die Vorfahren der Arten, die sich im entsprechenden Teilbaum befinden. Der Wert eines inneren Knotens gibt die Zeit an, die seit der Aufspaltung dieser Art vergangen ist. Von zwei inneren Knoten auf einem Pfad von der Wurzel zu einem Blatt ist immer einer der Vorfahre des anderen, wobei alle inneren Knoten und Blätter Nachkommen der Wurzel sind. Die Zeit seit Aufspaltung des Vorfahren ist immer größer als die Zeit seit Aufspaltung des Nachkommens. Dies erfüllt ein ultrametrischer Baum, da die Werte der inneren Knoten auf einem Pfad von der Wurzel stetig abnehmen sollen.

Für ultrametrische Daten gilt die *Theorie der Molekularen Uhr*. Diese Theorie besagt, daß die Anzahl der akzeptierten Mutationen in einer Proteinsequenz pro Zeiteinheit konstant ist. Als akzeptierte Mutationen werden Mutationen bezeichnet, die die Funktionsfähigkeit des Proteins aufrechterhalten und damit das Wei-

terleben der Art nicht verhindern. Je mehr Mutationen man also in zwei Proteinen findet, desto mehr Zeit ist seit der Aufspaltung der jeweiligen Arten vergangen.

Angenommen die Anzahl der akzeptierten Mutationen in zwei Proteinen sei k. Dann sagt die Theorie der Molekularen Uhr, daß die Aufspaltung der Arten vor $k/2$ Zeiteinheiten stattgefunden hat. Dieses Distanzmaß erfüllt damit alle Bedingungen der Ultrametrik. Es sei bemerkt, daß die Theorie der Molekularen Uhr in vielen Fällen nicht zutrifft.

Additive Distanzen

Eine schwächere Bedingung für Distanzdaten ist die *Additivität*. Additive Daten bzw. eine additive Matrix D erfüllt das sogenannte *Vierpunktkriterium*, das besagt: *Für alle $X_1, X_2, X_3, X_4 \in \{1, \ldots, n\}$ gibt es eine Nummerierung A,B,C,D, so daß*

$$D(A, B) + D(C, D) \leq D(A, C) + D(B, D) = D(A, D) + D(B, C).$$

Es gilt: Wenn D ultrametrisch ist, dann ist D auch additiv. Der Umkehrschluß gilt allerdings i.A. nicht. Eine ultrametrische Matrix ist also ein Spezialfall einer additiven Matrix.

Definition 7.3 (additiver Baum) *Ein additiver Baum zur additiven n x n Distanzmatrix D erfüllt folgende Bedingungen:*

- *er hat mindestens n Knoten, wobei genau die n Knoten, zu denen nur eine Kante führt (Blätter), mit den Spalten von D beschriftet sind*
- *jede Kante hat ein Gewicht*
- *die Summe der Gewichte auf einem Pfad zwischen zwei Blättern i und j ist genau $D(i, j)$.*

Ein additiver Baum ist ungerichtet und im Allgemeinen ohne Wurzel. Er repräsentiert damit die Entfernungen zwischen zwei Arten (Blättern), gibt aber keine Richtung der Evolution an. Das heißt, es ist nicht ersichtlich, wer Vorfahre und wer Nachkomme ist.

Bsp.: Eine additive Distanzmatrix für 4 Arten A, B, C, D ist in Tabelle 7.1 gegeben. Der zugehörige additive Baum wird in Abbildung 7.2 gezeigt. Das Vierpunktkriterium ist erfüllt, da

$$D(A, B) + D(C, D) = 13 \leq D(A, C) + D(B, D) = 17 = D(A, D) + D(B, C).$$

	A	B	C	D
A	0	3	8	10
B		0	7	9
C			0	10
D				0

Tabelle 7.1: Eine additive Distanzmatrix

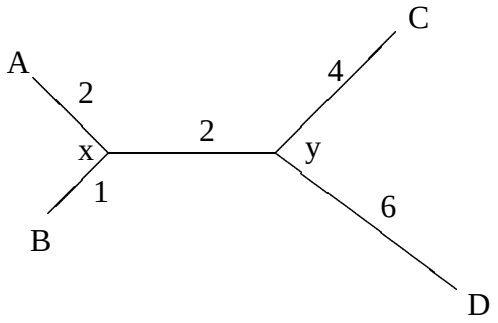


Abbildung 7.2: Additiver Baum zur Matrix in Tabelle 7.1

Für additive Daten gilt die Theorie der Molekularen Uhr im Allgemeinen nicht. Das bedeutet, zwei Arten können sich nach ihrer Aufspaltung mit unterschiedlicher Geschwindigkeit weiterentwickeln. So hat sich in Abbildung 7.2 A seit der Aufspaltung von x schneller entwickelt als B, Gleiches gilt auch für D und C.

Ein Algorithmus zur Konstruktion eines additives Baumes gegeben eine additive Distanzmatrix wird im folgenden Abschnitt besprochen.

Nicht additive Distanzen

Damit die Distanzdaten überhaupt auf einen Baum passen sollen, müssen sie additiv sein. Ansonsten ist nicht garantiert, daß es einen Baum gibt, dessen paarweise Kantenlängensummen den Distanzmatrixeinträgen entsprechen.

Auch für *nicht additive Daten* kann nach einem Baum suchen, der die Daten möglichst gut darstellt. Dazu versucht man beispielsweise den Baum zu finden, der eine minimale quadratische Abweichung der paarweisen Abstände zwei-

er Blätter i und j zu $D(i,j)$ hat [22]. Man minimiert also die Gleichung

$$\sum_{i,j \in \{1,\ldots,n\}} (D(i,j) - T(i,j))^2$$

wobei $T(i,j)$ die Summe der Kantengewichte des Pfades von i nach j ist. Wenn diese Summe 0 ist, hat man einen additiven Baum gefunden.

Allerdings ist diese Minimierung ein NP vollständiges Problem. Der Einsatz von Näherungsalgorithmen erlaubt aber das Finden einer suboptimalen Lösung.

7.2 Neighbor Joining

Neighbor Joining [51] ist ein iteratives Clusterungsverfahren, das gegeben eine additive Distanzmatrix D einen eindeutigen, additiven Baum findet. Neighbor Joining liefert als Ergebnis die Topologie des Baumes und seine Kantenlängen. Zu Beginn ist die Topologie ein Stern, bei dem jeder Knoten über zwei Kanten mit jedem anderen verbunden ist. In jeder Iteration werden 2 Arten zu einem Cluster zusammengefaßt, die neuen Kantenlängen anhand von D geschätzt und dies solange wiederholt, bis nur noch 2 Cluster übrig bleiben. Dabei bildet man den neuen Cluster aus den zwei Arten, die sowohl einen kleinen Abstand zueinander haben, als auch einen großen Abstand zu dem restlichen Baum.

Der Algorithmus besteht aus folgenden Schritten.

1. Berechne die Summe des Abstandes von Knoten i zu allen anderen Knoten für alle $i = 1, \ldots, n$
$$r_i = \sum_{k=1}^{n} D(i,k).$$

2. Finde aus allen $\binom{n}{2}$ Paaren das Paar (i,j), das
$$M_{ij} = D(i,j) - \frac{r_i + r_j}{n-2}$$
minimiert. $M(i,j)$ wird dabei klein, wenn $D(i,j)$ klein und r_i sowie r_j groß ist.

3. Ein neuer Knoten u verbindet jetzt das Paar (i,j) mit dem restlichen Baum. Die neuen Kantenlängen von i bzw. j zu u werden durch
$$s_{iu} = \frac{D(i,j)}{2} + \frac{r_i - r_j}{2(n-2)}$$

$$s_{ju} = \frac{D(i,j)}{2} + \frac{r_j - r_i}{2(n-2)}$$

berechnet. Nun müssen noch die neuen Kantenlängen von Knoten u zu jedem anderen Blatt $k \neq i, j$

$$s_{ku} = \frac{D(i,k) + D(j,k) - D(i,j)}{2}$$

berechnet werden.

4. Die Knoten i und j werden von der Liste aller Knoten gestrichen, dafür wird der neue Knoten u hinzugefügt. Damit verringert sich n um 1.

5. Wenn $n > 2$ beginnt man wieder bei Punkt 1. Ansonsten terminiert der Algorithmus. Die letzte Kante zwischen den beiden verbleibenden Knoten bekommt die Länge $s_{ij} = D(i,j)$.

Neighbor Joining hat eine Laufzeit von $O(n^3)$.

Der Beginn und die erste Iteration sind in Abbildung 7.3 dargestellt. Am Anfang sind alle n Knoten als Stern angeordnet. In der ersten Iteration werden die Knoten 2 und 3 zusammengefaßt, wodurch sich ein neuer Knoten u ergibt und die verbleibende Knotenanzahl um 1 reduziert.

In der Praxis sind Distanzdaten eigentlich nie additiv (und damit auch nicht ultrametrisch). Oftmals geht man aber von Additivität aus, um einen Baum in Polynomialzeit zu erhalten. Die paarweisen Kantenlängensummen des entstehenden Baumes werden dann allerdings nicht exakt den Distanzmatrixeinträgen entsprechen. Man hat dabei die Hoffnung, daß der entstehende Baum trotzdem die Distanzmatrix gut repräsentiert, obwohl nicht garantiert ist, daß es einen sinnvollen Baum zu einer nicht additiven Matrix gibt. Falls es genügend Evidenz dafür gibt, daß die Distanzdaten die Zeit seit Aufspaltung schätzen, kann man auch Ultrametrik annehmen.

Um dies zu verdeutlichen, wird in Tabelle 7.2 eine nicht additive Matrix gezeigt. Neighbor Joining findet dazu einen Baum (Abb. 7.4), dessen paarweise Kantenlängensummen etwas von den Distanzmatrixeinträgen abweichen. So ist die Kantenlängensumme von A zu C 8.75, während $D(A,B) = 9$. Außer für $D(A,B)$ und $D(C,D)$ gibt es für jedes Knotenpaar eine kleine Abweichung der Kantensumme zum Distanzmatrixeintrag.

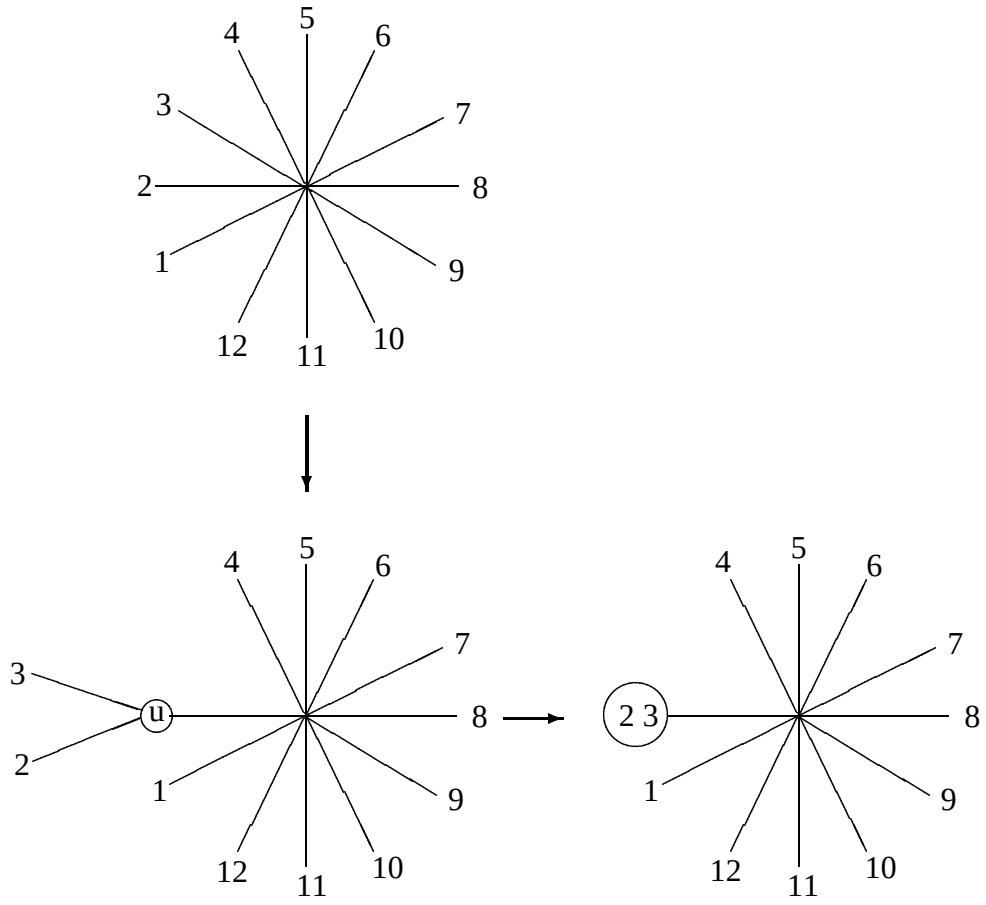


Abbildung 7.3: Neighbor Joining Schema

7.3 Ermittlung von Distanzdaten

Es gibt viele Möglichkeiten eine Distanz zweier Organismen zu bestimmen. Zum einen kann man Merkmale, wie “hat Flügel” oder “legt Eier” vergleichen und definiert dann die Distanz als Anzahl der unterschiedlichen Merkmale geteilt durch deren Gesamtzahl. Exaktere Ergebnisse erhält man, wenn man sich auf die Ebene der Genetik begibt. Hier kann man die Zahl der ähnlichen Gene oder die Reihenfolge von bestimmten Genen im Genom benutzen [32].

Weit verbreitet ist allerdings das *Alignment* von DNA oder Proteinsequenzen, das benutzt werden kann, um die Anzahl der akzeptierten Mutationen (siehe

	A	B	C	D
A	0	3	9	11
B		0	7	10
C			0	6
D				0

Tabelle 7.2: Eine nicht additive Distanzmatrix

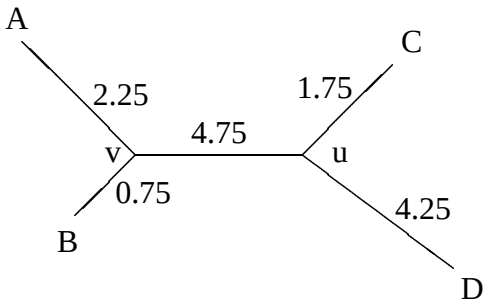


Abbildung 7.4: Neighbor Joining Baum zur Matrix in Tabelle 7.2

Kap. 7.1) zweier Sequenzen zu schätzen. Dabei wird ein inexakter Mustervergleich zwischen konkreten Genomabschnitten oder Aminosäuresequenzen durchgeführt. Das Alignment liefert eine Möglichkeit, wie eine Sequenz unter Verwendung von Insertionen, Deletionen und Ersetzung von Zeichen in eine andere umgewandelt werden kann. Je mehr Edit Operationen notwendig sind, um so wahrscheinlicher ist es, daß die Organismen, deren Sequenzen man betrachtet, sich vor langer Zeit aufgespalten haben, wenn man eine konstante Mutationsrate voraussetzt. Man benutzt also Mechanismen der Evolution, um den Grad der Verwandtschaft zweier Arten zu bewerten. Alignment liefert daher eine Schätzung der vergangenen Zeit seit Aufspaltung der Arten.

Ein $O(n^2)$ Algorithmus für exaktes, paarweises Sequenzalignment wurde 1970 von Needleman und Wunsch [41] vorgestellt, auf dem viele, heute verwendete Verfahren basieren. Unterschiede gibt es in der Bewertung der einzelnen Edit Operationen. Alignment kann auch auf mehrere Sequenzen erweitert werden (*multiples Alignment*), was zur Bestimmung der Distanzen zwischen mehreren Arten benutzt wird.

Ein Beispiel für ein Alignment von zwei DNA Sequenzen ist in Abbildung 7.5 dargestellt. Waagerechte Striche in der oberen Sequenz stellen Deletionen dar, Striche in der unteren Sequenz Insertionen von Basen in die obere Sequenz. Übereinstimmungen werden durch senkrechte Striche dargestellt, die bei Mismatches fehlen. Dieses Alignment liefert 5 Edit Operationen, die die obere Sequenz in die untere überführen. Die Anzahl der akzeptierten Mutationen würde man daher auf 5 schätzen.

```
C G A T A - T - G A C A A C T G
|   | | |   |   | |   | |   | |
C T A T A A T C G A - A A G T G
```

Abbildung 7.5: Alignment zweier DNA Sequenzen

Allerdings ist Alignment nicht in der Lage mit evolutionären Ereignissen, wie Duplikationen oder umgekehrten Komplementen umzugehen. Deshalb ist es nur sinnvoll Alignment auf stark konservierte Sequenzen, wie verwandte Gene oder Proteine, welche eine ähnliche Funktion haben, anzuwenden.

Mit Hilfe *bedingter Kompression* kann ein Kompressionsalgorithmus diese Nachteile umgehen. Bei der bedingten Kompression komprimiert man eine Sequenz gegeben eine Referenzsequenz. *GenCompress* beispielsweise kodiert dann auch Wiederholungen, deren Quellvorkommen sich in der Referenzsequenz befinden. Bei *CDNA* würde man die Referenzsequenz als Trainingssequenz betrachten. Wenn beide Sequenzen sehr ähnlich oder im Idealfall gleich sind, dann ergibt sich eine sehr hohe Kompressionsrate. Im umgekehrten Fall, wenn die Sequenzen komplett verschieden sind, ist die Kompressionsrate sehr niedrig oder negativ. Die Kompressionsrate ist ein Maß für die Ähnlichkeit zweier Sequenzen.

Ein Kompressionsalgorithmus ist nicht nur beschränkt auf einzelne Gene, sondern kann auf ganze Genome angewendet werden. Weiterhin kann ein Kompressor mit Wiederholungen und umgekehrten Komplementen umgehen. Wenn große Teile der beiden Sequenzen ein umgekehrtes Komplement sind, dann wird der Kompressor dies erkennen und eine hohe Kompressionsrate erreichen. Ein Alignment hingegen wird in diesem Fall nur wenig Evidenz für eine enge Verwandtschaft der Sequenzen liefern.

Die Sequenzierung von kompletten Genomen schreitet heute rasch voran und ist weitgehend automatisch durchführbar. Welche Gene eine Sequenz enthält, welche Proteine diese Gene kodieren oder die Bestimmung der Primärstruktur von

Proteinen ist weitaus schwieriger und erfordert zum großen Teil menschliche Interaktion. Daher sind oftmals die DNA Sequenzen von Organismen bekannt, deren Gene oder Proteine aber nur zu einem kleinen Teil. Die Verwandtschaftsbestimmung anhand der Genome kann dabei vollautomatisch stattfinden, da es im Gegensatz zu Alignment–basierten Verfahren nicht notwendig ist, Gene oder Proteinsequenzen zu bestimmen. Darüber hinaus können auch nichtkodierende Abschnitte im Genom berücksichtigt werden.

Ein Abstandsmaß d, das die Distanz der Sequenzen x und y angibt, muß folgende Bedingungen erfüllen:

1. $d(x, y) > 0$ für $x \neq y$
2. $d(x, x) = 0$
3. $d(x, y) = d(y, x)$ (Symmetrie)
4. $d(x, y) \leq d(x, z) + d(z, y)$ (Dreiecksungleichung).

Für ultrametrische und additive Distanzen sind diese Bedingungen erfüllt.

Im Gegensatz zum perfekt symmetrischen Alignment ist ein Distanzmaß, das allein auf der bedingten Kompression basiert, stark unsymmetrisch [36, 24]. Das ist problematisch, da unklar ist, ob man als Abstand von x zu y $d(x, y)$ oder $d(y, x)$ verwenden soll.

Mit Hilfe eines Theorems aus der Theorie der Kolmogorov Komplexität ist es aber möglich, aus der bedingten Kompression ein Distanzmaß zu konstruieren, das diesen 4 Eigenschaften genügt.

7.4 Kolmogorov Komplexität

Die *Kolmogorov Komplexität* beschreibt die Komplexität eines gegebenen Objektes durch die Länge des kürzesten Programms, das dieses Objekt ausgibt. Dies ist äquivalent zur absolut notwendigen Bitanzahl, die man für eine vollständige Beschreibung des Objektes benötigt. Die Kolmogorov Komplexität $K(x)$ eines Objektes x ist dabei unabhängig von der verwendeten Programmiersprache. Jede universelle Programmiersprache liefert bis auf die Addition einer Konstante die gleiche Komplexität. Je komplexer ein Objekt ist, desto größer ist sein Informationsgehalt und desto größer wird auch $K(x)$.

Die *bedingte Kolmogorov Komplexität* $K(x|y)$ ist die Länge des kürzesten Programms, das x ausgibt, wenn y als Input gegeben ist. $K(x)$ kann somit auch

als $K(x|\epsilon)$ betrachtet werden, wobei ϵ das leere Objekt ist. $K(x|y)$ ist also die verbleibende Information in x, wenn y bekannt ist.

Theorem 7.1 (Symmetrie der Information) *Mit einem additiven logarithmischen Faktor gilt:*

$$K(x|y) + K(y) = K(y|x) + K(x).$$

Dieses Theorem (Theorem 6.1 in [12]) besagt, daß $K(x) - K(x|y) \approx K(y) - K(y|x)$, d.h. die Information, die x über y hat, ist etwa gleich der Information, die y über x hat.

Li et al. [32] schlagen als Distanzmaß

$$d(x, y) = 1 - \frac{K(x) - K(x|y)}{K(xy)}$$

vor. Dabei ist $K(x) - K(x|y)$ die Menge an Information, die y über x weiß. $K(xy)$ ist die Kolmogorov Komplexität von x verkettet mit y und dient als Normalisierungsfaktor, da bis auf eine additive Konstante $K(x|y) \leq K(x)$ und $K(xy) \geq K(x)$ gilt. Damit liegt $d(x, y)$ zwischen 0 und 1. Dabei bedeutet $d(x, y) = 0$, daß y alles über x weiß und $d(x, y) = 1$, daß y nichts über x weiß.

Dieses Abstandsmaß erfüllt die 4 Bedingungen einer Distanz. Das Theorem 7.1 stellt die Symmetrie $d(x, y) = d(y, x)$ sicher. Die zweite Bedingung ist leicht nachzurechnen, da $K(x|x) = 0$ und $K(xx) = K(x)$ und auch die erste ist erfüllt. Weiterhin wird in [32] (Theorem 1) die Dreiecksungleichung bewiesen.

Die Kolmogorov Komplexität ist allerdings nicht berechenbar. Deshalb muß eine Näherung für $K(x)$, $K(x|y)$ und $K(xy)$ gefunden werden. Da $K(x)$ den Informationsgehalt oder die komprimierte Größe von x angibt, kann ein Kompressor eine obere Schranke von $K(x)$ und $K(xy)$ berechnen. Die bedingte Kompression liefert eine Näherung für $K(x|y)$.

7.5 Rekonstruktion von Phylogenien

Von Li et al. [32] wurde gezeigt, daß trotz der Approximation der Kolmogorov Komplexität ein Kompressor geeignet ist, um aus vollständigen Genomen die Phylogenie von Organismen zu rekonstruieren. Dazu hat Li et al. folgenden Versuch unternommen. Es wurden die Mitochondriengenome von 20 Säugetieren betrachtet und für alle Paare die Distanz $d(x, y)$ berechnet, indem *GenCompress* zur approximativen Berechnung von $K(x)$, $K(x|y)$ und $K(xy)$ benutzt wurde. Die mit

GenCompress berechneten Distanzen sind zwar nicht perfekt symmetrisch, aber die Unterschiede sind so gering, daß sie vernachlässigt werden können. Aus der resultierenden Distanzmatrix wurde mit Neighbor Joining ein phylogenetischer Baum erstellt. Es konnte gezeigt werden, daß dieser Baum identisch zu einem Stammbaum ist, der auf dem Alignment von Mitochondrienproteinen basiert.

Um die Leistungsfähigkeit des Verfahrens bei heterogeneren und größeren Datensätzen zu untersuchen, habe ich folgenden Versuch durchgeführt. Mit Hilfe von *GenCompress* sollte die Phylogenie von 41 Bakterien anhand ihrer kompletten Genome rekonstruiert werden. In Tabelle 7.3 sind alle Bakterien, deren GenBank Locus und die Länge des Genoms aufgelistet.

Die kompletten Genome sind in *GenBank* unter [5] verfügbar. Für alle 820 Paare von Bakterien habe ich mit der *GenCompress* Implementierung [28] den Abstand $d(x, y)$ berechnet. Auf einem Rechner mit Intel Pentium III 450 MHz CPU und 256 MB RAM benötigt man zwischen 1 und 10 Stunden für die Berechnung einer paarweisen Distanz, was von der Länge der beiden Genome abhängt. Um Rechenzeit zu sparen, habe ich nur das obere Dreieck der Distanzmatrix berechnet.

Zur Erstellung des phylogenetischen Baumes wurde eine Implementierung der Neighbor Joining Methode (Programm *neighbor*), die Bestandteil des umfangreichen *Phylip Package* [19] ist, verwendet. Das Ergebnis ist in Abbildung 7.6 gezeigt, wobei beim Plotten des Baumes die Kantenlängen nicht berücksichtigt wurden. Zum direkten Vergleich wurde ein Baum herangezogen, der wiederum auf dem Alignment von Proteinsequenzen basiert und in Abbildung 7.8 dargestellt ist (mit freundlicher Einwilligung von Arndt von Haeseler). Dieser Baum enthält außerdem zwei Pilze (Candida albicans und Saccharomyces cerevisiae), die in meinem Datensatz nicht vertreten waren. Beide Bäume weisen wesentliche Gemeinsamkeiten auf. So findet man z.B. die Gruppe der *Archaebakterien* (Aeropyrum, Halobacterium, Pyrococcus a., Pyrococcus h., Archaeoglobus, Methanococcus, Methanothermobacter, Thermoplasma a., Thermoplasma v.) vollständig in einem Teil des Baumes wieder, wobei allerdings zwei weitere Bakterien (Thermotoga, Aquifex) wahrscheinlich nicht korrekt den Archaebakterien zugeordnet wurden. Bis auf Ausnahmen stimmt die Topologie zwischen beiden Bäumen überein.

Abbildung 7.7 zeigt den Baum, bei dessen Plot die Kantenlängen mit berücksichtigt wurden. Hier werden die Grenzen von Kompressionsverfahren bei der Rekonstruktion von Phylogenien deutlich. Nur die Neisseria und Helicobacter Arten sollten sich in evolutionär jüngerer Zeit aufgespalten haben. Alle anderen Bakterien wären zu etwa gleicher Zeit aus einer Art hervorgegangen; eine Aussage,

Organismus	**GenBank Locus**	**Größe (bp)**
Aeropyrum pernix	BA000002	1669695
Aquifex aeolicus	AE000657	1551335
Archaeoglobus fulgidus	AE000782	2178400
Bacillus halodurans	BA000004	4202353
Bacillus subtilis	AL009126	4214814
Borrelia burgdorferi	AE000783	910724
Buchnera sp. APS	AP000398	640681
Campylobacter jejuni	AL111168	1641481
Caulobacter crescentus	AE005673	4016947
Chlamydia trachomatis	AE001273	1042519
Chlamydophila pneumoniae	AE001363	1230230
Deinococcus radiodurans	AE000513	2648638
Escherichia coli K12	U00096	4639221
Escherichia coli O157:H7	AE005174	5528445
Haemophilus influenzae Rd	L42023	1830138
Halobacterium sp. NRC-1	AE004437	2014239
Helicobacter pylori 26695	AE000511	1667867
Helicobacter pylori J99	AE001439	1643831
Lactococcus lactis subsp. lactis	AE005176	2365589
Mesorhizobium loti	BA000012	7036074
Methanococcus jannaschii	L77117	1664970
Methanothermobacter thermautotrophicus	AE000666	1751377
Mycobacterium leprae	AL450380	3268203
Mycobacterium tuberculosis	AL123456	4411529
Mycoplasma genitalium	L43967	580074
Mycoplasma pneumoniae	U00089	816394
Neisseria meningitidis	AE002098	2272351
Neisseria meningitidis Z2491	AL162759	2184406
Pasteurella multocida	AE004439	2257487
Pseudomonas aeruginosa	AE004091	6264403
Pyrococcus abyssi	AL096836	1765118
Pyrococcus horikoshii	BA000001	1738505
Rickettsia prowazekii	AJ235269	1111523
Streptococcus pyogenes	AE004092	1852441
Synechocystis PCC6803	AB001339	3573470
Thermoplasma acidophilum	AL139299	1564906
Thermoplasma volcanium	BA000011	1584804
Thermotoga maritima	AE000512	1860725
Treponema pallidum	AE000520	1138011
Ureaplasma urealyticum	AF222894	751719
Vibrio cholerae	AE003852	2961149
Xylella fastidiosa	AE003849	2679306

Tabelle 7.3: Daten der verwendeten 41 Bakterien

die sehr unwahrscheinlich ist. Offenbar ist *GenCompress* bei zu unterschiedlichen Genomen nicht mehr in der Lage, die tatsächliche Ähnlichkeit exakt anzugeben, so daß auch bei nah verwandten Arten die Distanz ≈ 1 ist (Bsp.: $d(\text{Mycoplasma genitalium, Mycoplasma pneumoniae}) = 0.9942$). Alignment von Proteinsequenzen liefert hier viel genauere Werte und kann die evolutionären Zeitpunkte besser schätzen. Trotzdem ist es erstaunlich, daß sich die grobe Struktur des Baumes aus den offensichtlich sehr kleinen Distanzunterschieden recht gut rekonstruieren läßt.

Zum Vergleich des Säugetierdatensatzes von Li et al. und den von mir verwendeten 41 Bakterien muß Folgendes erwähnt werden. Die Aufspaltung der Säugetiere begann vor etwa 150 Mill. Jahren, die Aufspaltung der Bakterien vor etwa 1 Mrd. Jahren, also vor einem mehr als sechsmal längeren Zeitraum.

Bei zu heterogenen Genomen ist es *GenCompress* nicht mehr möglich, kleine Gemeinsamkeiten zu erkennen. Um die Grenzen dieser Methode zu untersuchen, habe ich anhand von konstruierten Sequenzen mit bekanntem Abstand, die *GenCompress*–Distanz berechnet und mit der wahren Distanz verglichen. Dazu habe ich zuerst eine Sequenz von 1 Mill. bp zufällig generiert und anschließend $1, 2, 3, \ldots, 15, 20, 25, \ldots, 50, 60, 70, \ldots, 100$ Prozent an willkürlich gewählten Basen verändert. Für jede mutierte Sequenz wurde der Abstand zur Ausgangssequenz berechnet.

Das Ergebnis ist in Abbildung 7.9 grafisch dargestellt. Da die Distanz der Sequenzen der Prozentzahl an mutierten Positionen entspricht, müßte bei perfekter Rekonstruktion der paarweisen Sequenzabstände eine Gerade $y = x$ zu sehen sein. Tatsächlich ergibt sich ein ganz anderes Bild. Schon ab etwa 25 Prozent an veränderten Basen erkennt *GenCompress* fast keine Gemeinsamkeit der beiden Sequenzen mehr. Die beiden Neisseria Arten haben in der Distanzmatrix einen Wert von 0.384, die beiden Helicobacter Arten einen Wert von 0.563. Das bedeutet, ihre Genome haben wahrscheinlich eine Ähnlichkeit von weit über 75%. Für alle anderen Paare von Bakterien läßt sich die Ähnlichkeit ihrer Genome nicht mehr genau schätzen, da sie alle in der Distanzmatrix einen Wert haben, der nahe bei 1 ist. Auch wenn in diesem Versuch nur Ersetzungen von Basen und keine anderen Evolutionsereignisse, wie Insertionen, Deletionen, Duplikationen berücksichtigt wurden, wird doch deutlich, warum *GenCompress* keine genaue Distanzschätzung bei stark unterschiedlichen Sequenzen liefern kann.

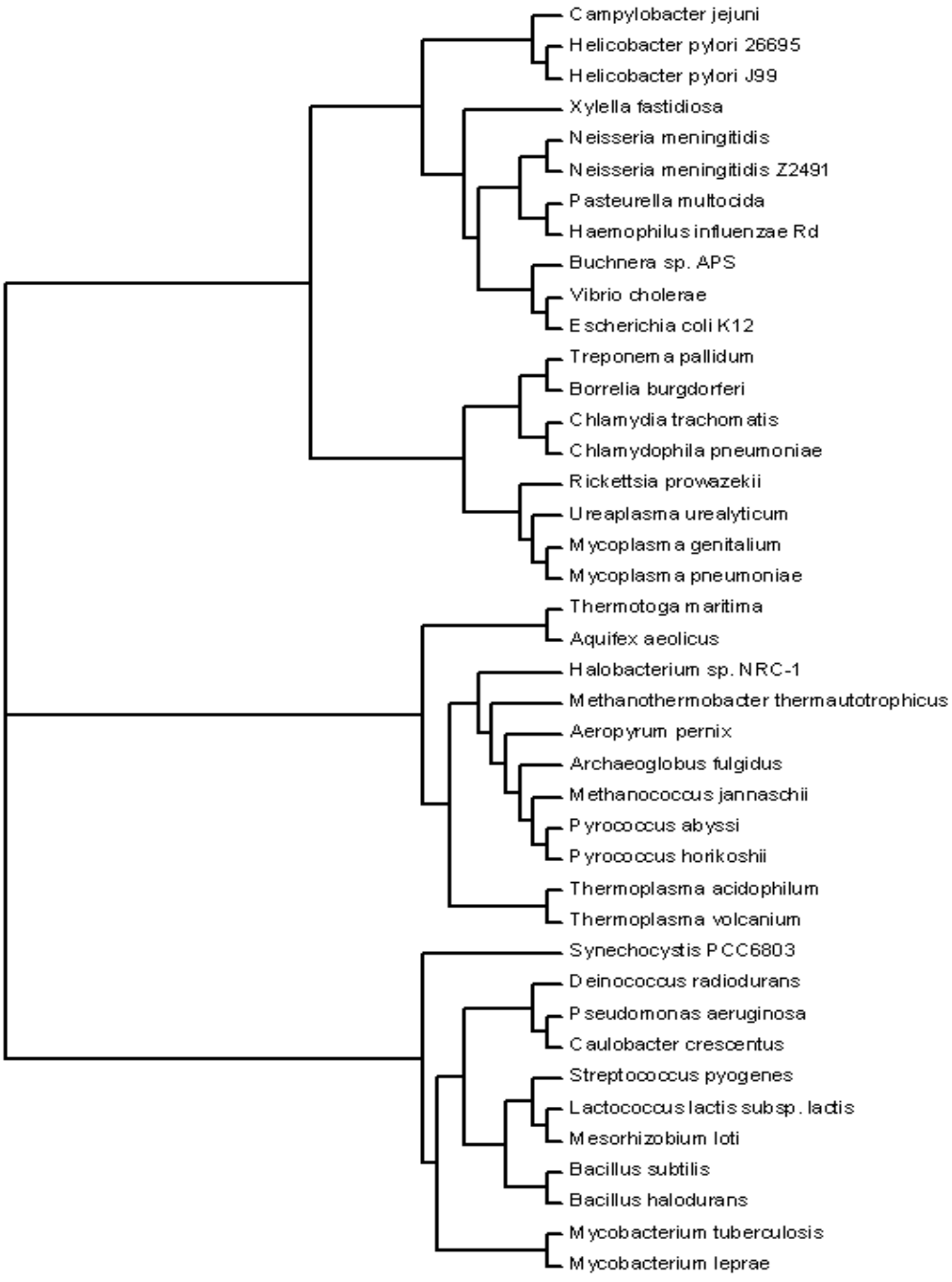


Abbildung 7.6: Phylogenetischer Baum der 41 Bakterien ohne Kantenlängen

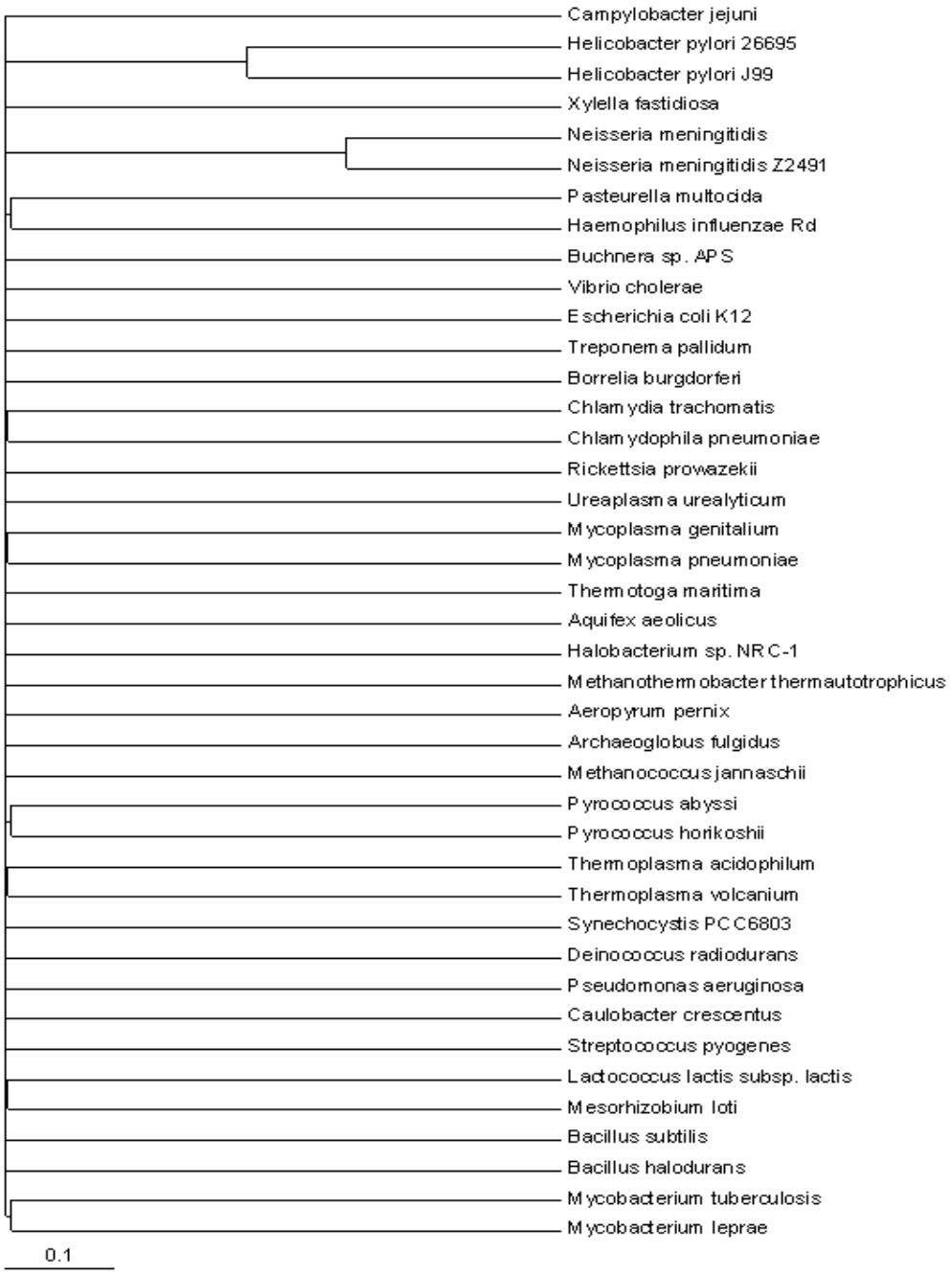


Abbildung 7.7: Phylogenetischer Baum der 41 Bakterien mit Kantenlängen

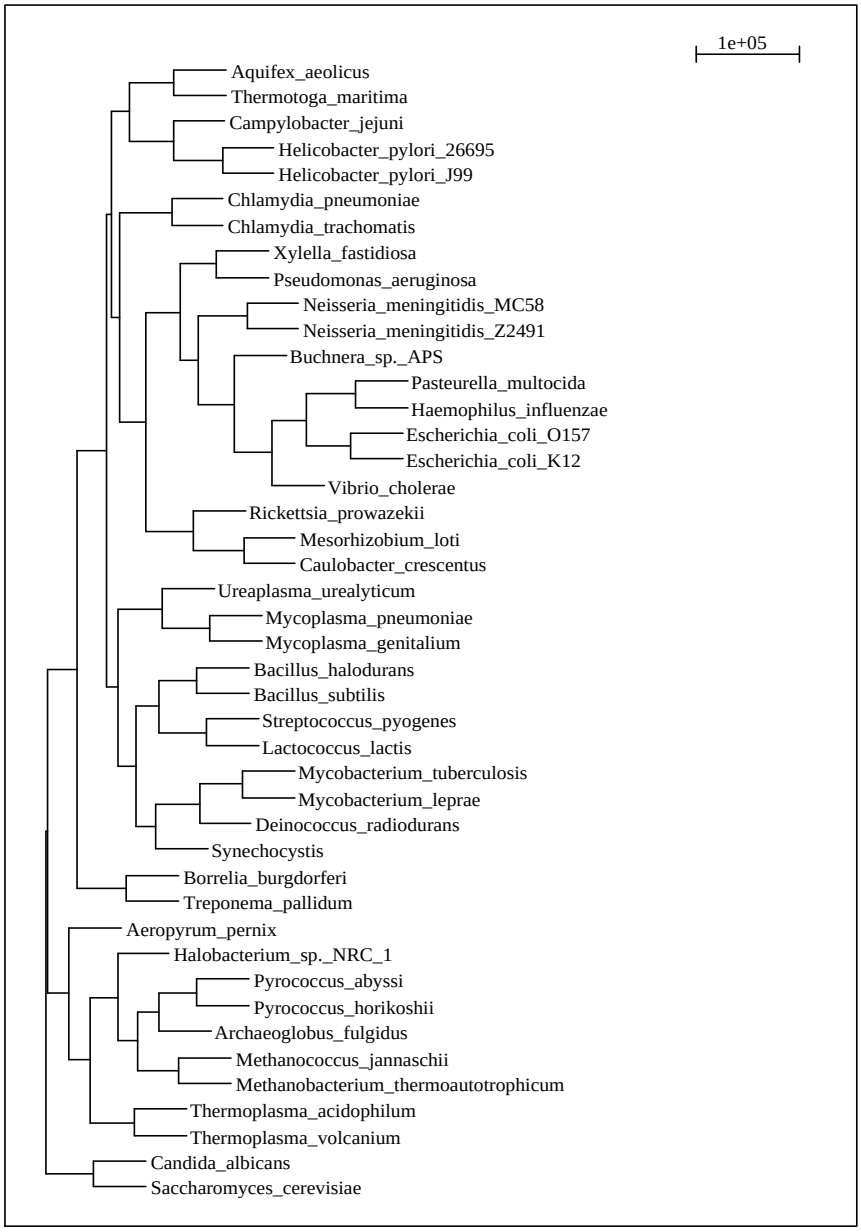


Abbildung 7.8: Phylogenetischer Baum mit Kantenlängen basierend auf Alignment von Proteinsequenzen (Quelle: Arndt von Haeseler, unpublished)

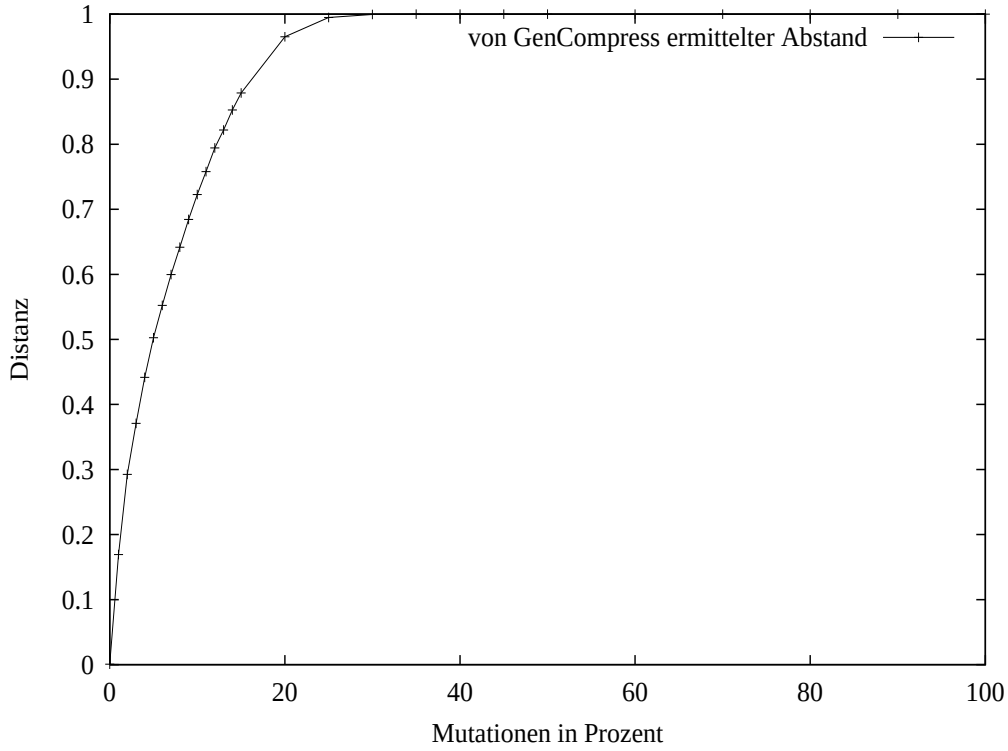


Abbildung 7.9: GenCompress Distanz

7.6 Zusammenfassung

Zusammenfassend läßt sich sagen, daß mit bedingter Kompression und dem Distanzmaß d ein vollautomatisches Verfahren zum Vergleich von kompletten Genomen zur Verfügung steht. Diese Methode hat einige Vorteile gegenüber dem Alignment, da evolutionäre Ereignisse, wie Wiederholungen und umgekehrte Komplemente mit berücksichtigt werden.

Die bedingte Kompression kann angewendet werden, um Phylogenien zu rekonstruieren. Dafür berechnet man die Distanz für alle Paare von Genomen. Bei wenig konservierten Genomen, deren Aufspaltung vor evolutionär langer Zeit stattfand, kann allerdings höchstens die Topologie des Baumes rekonstruiert werden, da *GenCompress* bei Genomen, die sich in mehr als 25 % ihrer Basenpaare unterscheiden, fast keine Ähnlichkeit mehr aufdecken kann.

Um Aussagen über die Zeiten der Aufspaltung zu machen, muß man sich auf konservierte Sequenzen und kurze Evolutionszeiträume beschränken. Im Gegensatz zu Alignment ist aber keine aufwendige Suche nach Genen oder Proteinen nötig. Die in nichtkodierenden Genomabschnitten enthaltene Information über den evolutionären Abstand wird ebenfalls mit berücksichtigt.

Die Anwendbarkeit der Methode ist weiterhin auf kürzere Genome (kleiner als 10 Millionen bp), wie Bakterien– oder Mitochondriengenome, beschränkt, da sonst die Ressourcen und Rechenkapazität eines normalen Rechners nicht mehr ausreichen.

Kapitel 8

Entropieunterschiede zwischen Exons und Introns

Dieses Kapitel beschäfigt sich mit einer genauen Untersuchung der Entropieunterschiede zwischen Exons und Introns. Dabei wird zuerst erläutert, warum ein Entropieunterschied zu erwarten ist. Danach werde ich bisherige Arbeiten zu diesem Thema vorstellen und deren Ergebnisse an größeren Datensätzen verschiedener Organismen verifizieren. Anschließend wird untersucht, in wie weit sich diese Ergebnisse bei der Vorhersage von Exons verwenden lassen.

8.1 Motivation und bisherige Arbeiten

In Eukaryoten bestehen viele Gene nicht nur aus einem einzigen, kodierenden Abschnitt, sondern sind durch nichtkodierende Stücke (Introns) unterbrochen. Bevor das Primärtranskript eines Gens in das entsprechende Protein translatiert werden kann, müssen die Introns entfernt werden, was als Spleissen bezeichnet wird. Das Ergebnis ist eine Kette aller Exons des Gens.

Während Exons die genetische Information tragen, ist die Funktion der Introns (sofern es eine gibt) noch weitgehend unbekannt. Es ist zwar bekannt, daß Introns Regionen enthalten können, die die Genexpression steuern, aber dies trifft wahrscheinlich nur auf einen kleinen Teil aller Introns zu.

Zufällige Mutationen sollten in Exon- und Intronabschnitten gleichhäufig vorkommen, aber bei Exons sollte die Auswirkung viel dramatischer sein, da hier das Leseraster durch Insertionen und Deletionen verschoben werden kann oder Codons verändert werden können. Da Introns keine Information über das Prote-

in kodieren, erwartet man, daß Mutationen in diesen Abschnitten häufig erhalten bleiben, während durch einige Mutationen in Exons ein funktionsunfähiges Protein entstehen kann. Wenn dieses Protein lebensnotwendig ist, sollte der Träger der Mutation aussterben und damit sein verändertes genetisches Material nicht weitergeben können. Nach evolutionär langer Zeit sollten sich in Introns viele Mutationen angesammelt haben, in Exons dagegen nur wenige. Dadurch sollten Introns einen zufälligeren Charakter als Exons haben.

In der Informationstheorie würde man diese Vermutung durch zwei unterschiedliche Quellen modellieren. Dabei erzeugt die Intron–Quelle zufälligere Sequenzen als die Exon–Quelle. Zwischen beiden Quellen sollte sich dann ein Entropieunterschied ergeben. Da Introns einen zufälligeren Charakter als Exons haben sollten, würde man erwarten, daß die Intron–Quelle eine etwas höhere Entropie besitzt.

Introns und Exons sind meistens sehr kurz. Menschliche Exons sind im Durchschnitt 221 bp, Introns 741 bp lang. Da einige Exons und Introns sehr lang sind, während ihre Mindestlänge bei etwa 30 bp liegt, sind die medianen Längen noch einmal bedeutend kürzer (Exons 139 bp, Introns 354 bp). Diese Werte ergeben sich aus einem Datensatz, dessen Erstellung ich weiter unten erläutern werde, und sollen nur einen ungefähren Eindruck von der Länge vermitteln. Für andere Organismen ergeben sich ähnliche Werte. Um die Entropie von diesen kurzen Sequenzen so genau zu schätzen, daß ein eventuell kleiner Entropieunterschied aufgedeckt werden kann, benötigt man einen schnell konvergierenden Entropieschätzer. Diese Eigenschaft erfüllt der Match Length Entropieschätzer, der in Kapitel 6.1.3 beschrieben wurde.

Farach et al. [21] wendete 1994 den Match Length Entropieschätzer erfolgreich an, um den vermuteten Entropieunterschied nachzuweisen. Aus der Nukleotiddatenbank *GenBank* wurden 669 Exons und 659 Introns aus 66 vollständig annotierten Genen extrahiert. Für ein Exon wurde die Entropie geschätzt und diese anschließend mit der Entropie des benachbarten Introns verglichen. Die Schätzung ist durch

$$\hat{H} = \frac{log_2 n}{\overline{L}}$$

gegeben, wobei $\overline{L}$ der Mittelwert der Matchlängen ist. Zur Berechnung der Matchlängen wurde das längste Präfixmatch mit den 16 zurückliegenden Zeichen ($n = 16$) gesucht.

Als Ergebnis hätte man erwartet, daß die Mehrheit der Exons eine kleinere Entropie als ihr benachbartes Intron haben. Überraschenderweise ergab sich bei dem Versuch von Farach et al., daß 73 % der Introns eine niedrigere Entropie

als das Nachbarexon haben. Dieses Ergebnis ist außerdem statistisch signifikant (Mann–Whitney–U Test, p–Wert von 10^{-5}).

Meiner Meinung nach könnte es mehrere Erklärungen geben, warum im Gegensatz zu den obigen Überlegungen Exons keine niedrigere Entropie haben. Zuerst ist es möglich, daß die Entropie als Maß nicht gut geeignet ist, um die unterschiedlichen Eigenschaften von Exons und Introns zu charakterisieren. Eine andere Möglichkeit sind bisher nicht verstandene Funktionen der Introns. Diese Funktionen könnten für eine korrekte Transkription essentiell sein, so daß auch Introns restaurativen Kräften unterliegen. So ist z.B. bekannt, daß Mutationen an Spleisstellen ein Spleissen des Introns verhindern können, was durch ein verschobenes Leseraster und zusätzliche Codons starke Auswirkungen auf ein Protein haben sollte. Weiterhin gibt es auch bei Exons Positionen an denen eine Mutation noch nicht einmal die Primärstruktur des Proteins verändert. Die Tabelle des genetischen Kodes (Seite 11) zeigt, daß die dritte Codonposition oftmals redundant ist. Selbst wenn eine Mutation im Exonabschnitt das Codon so verändert, daß eine andere Aminosäure in die Proteinsequenz eingefügt wird, muß das Protein deshalb nicht seine Funktionsfähigkeit verlieren. Die Ersetzung einer Aminosäure durch eine chemisch ähnliche liefert in den meisten Fällen dieselbe 3D Struktur, so daß zwei mutierte Gensequenzen durchaus zu einem Protein führen können, das voll funktionsfähig ist. Außerdem ist es möglich, daß Mutationen nicht zufällig sind bzw. daß es Mutationen gibt, die bevorzugt auftreten. Das könnte auch erklären, warum Introns, die mehr Mutationen enthalten sollten, weniger zufällig erscheinen.

Der Match Length Entropieschätzer kann außer dem Mittelwert der Matchlänge $\overline{L}$ auch dessen Varianz bestimmen, die im Folgenden als *Matchlängenvarianz (ML–Varianz)* bezeichnet wird. Die Matchlängenvarianz berechnet sich als $\overline{L^2} - \overline{L}^2$, wobei $\overline{L^2}$ der Mittelwert der quadrierten L_i ist. Für Regionen mit niedrigerer Entropie sollte diese Varianz einen höheren Wert liefern. Farach et al. haben auch die Matchlängenvarianz von benachbarten Exons und Introns verglichen. Wie erwartet, hatten die Mehrheit der Introns (80 %) eine höhere ML–Varianz als das Nachbarexon. Dabei erweist sich die ML–Varianz als ein noch stärkerer Diskriminator im Vergleich zur Entropie.

8.2 Untersuchung an größeren Datensätzen

Als Farach et al. 1994 das oben beschriebene Experiment durchführten, waren nur wenige menschliche Gene bekannt, so daß der verwendete Datensatz viel-

leicht nicht repräsentativ für das menschliche Genom sein könnte. Mittlerweile sind einige tausend menschliche Gene vollständig annotiert. Weiterhin ist es interessant zu überprüfen, ob man auch bei anderen Organismen einen Unterschied der Entropie und Matchlängenvarianz nachweisen kann. Diese Aufgabe ist ein wichtiger Teil dieser Diplomarbeit und soll jetzt detailliert besprochen werden.

Datenbeschaffung

Zur Beschaffung der Daten wurde wieder die Nukleotiddatenbank *GenBank* [17] verwendet. Um die Entropien und ML–Varianzen von benachbarten Exons und Introns zu berechnen, braucht man die vollständige Sequenz eines Gens und eine Liste der Start– und Endpositionen aller enthaltenen Exons. Da *GenBank* kein relationales oder objektorientiertes Datenbankschema verwendet, sondern nur Dateien verwaltet, ist es nicht möglich, eine solche Liste mit einer entsprechenden Datenbankanfrage zu erhalten. Um Features aus einem solchen *GenBank* Flatfile zu extrahieren, ist es notwendig, sich einen eigenen Parser zu schreiben. Ein typisches *GenBank* Flatfile eines menschlichen Gens ist im Folgenden gezeigt. Ein *GenBank* Eintrag beginnt mit allgemeinen Informationen, wie einem Bezeichner (*LOCUS*), der Länge und Herkunft der Sequenz, dem Organismus, einer kurzen Beschreibung (*DEFINITION*) und Literaturverweisen. Danach schließt sich die Feature Tabelle an, eingeleitet durch das Schlüsselwort *FEATURES*. Nun folgt eine Liste von bekannten Merkmalen der Sequenz, wie transkribierten Bereichen (*mRNA*) und Exon–, Intronpositionen (*exon, intron*). Mit dem Schlüsselwort *ORIGIN* beginnt die eigentliche Sequenz, die mit \\ endet.

```
LOCUS       HUMDES                 11990 bp    DNA     linear   PRI 22-MAR-2001
DEFINITION  Homo sapiens desmin gene, complete cds.
ACCESSION   M63391 M26935 M58168 M59379 M65071
VERSION     M63391.1  GI:181539
KEYWORDS    Alu repeat; desmin.
SOURCE      human.
  ORGANISM  Homo sapiens
            Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
            Mammalia; Eutheria; Primates; Catarrhini; Hominidae; Homo.
REFERENCE   1  (bases 1 to 11990)
  AUTHORS   Li,Z.L., Lilienbaum,A., Butler-Browne,G. and Paulin,D.
  TITLE     Human desmin-coding gene: complete nucleotide sequence,
            characterization and regulation of expression during myogenesis and
            development
  JOURNAL   Gene 78 (2), 243-254 (1989)
  MEDLINE   89378751
   PUBMED   2673923
REFERENCE   2  (bases 1 to 3280)
  AUTHORS   Li,Z.L. and Paulin,D.
  TITLE     High level desmin expression depends on a muscle-specific enhancer
```

```
  JOURNAL   J. Biol. Chem. 266 (10), 6562-6570 (1991)
  MEDLINE   91177917
COMMENT     Draft entry and computer readable copy of sequence kindly submitted
            by D.Paulin, 05-SEP-1989.
            The accession number M58186 is cited in reference J. Biol. Chem.
            266, 6562- 6570 (1991). The correct accession number is M58168.
FEATURES             Location/Qualifiers
     source          1..11990
                     /organism="Homo sapiens"
                     /db_xref="taxon:9606"
     enhancer        2224..2500
     misc_binding    3116..3129
                     /bound_moiety="MEF1"
     TATA_signal     3163..3168
                     /note="putative; putative"
     mRNA            join(3198..3852,4905..4965,5062..5157,5306..5467,
                     5646..5771,6157..6377,8555..8598,10425..10507,
                     10692..>11463)
                     /product="desmin"
     exon            3198..3852
                     /number=1
                     /product="desmin"
     CDS             join(3278..3852,4905..4965,5062..5157,5306..5467,
                     5646..5771,6157..6377,8555..8598,10425..10507,
                     10692..10733)
                     /codon_start=1
                     /product="desmin"
                     /protein_id="AAA99221.1"
                     /db_xref="GI:181540"
                     /translation="MSQAYSSSQRVSSYRRTFGGAPVFSLGSPLSSPVFPRAPFGSKG
                     SSSSVTSRVYQVSRTSGGAGGLGSLRASRLGTTRTPSSYGAGELLDFSLADAVNQEFL
                     TTRTNEKVELQELNDRSPIYMEKVRFLEQQNALAAEVNRLKGREPTRVAELYEEELRE
                     LRRQVEVLTNQRARVDVERDNLLDDLQRLKAKLQEEIQLKEEAENNLAAFRADVDAAT
                     LARIDLERRIESLNEEIAFLKKVHEEEIRELQAQLQEQQVQVEMDMSKPDLTAALRDI
                     RAQYETIAAKNISEAEEWYKSKVSDLTQAANKNNDALRQAKQEMMEYRHQIQSYTCEI
                     DALKGTNDSLMRQMRELEDRFASEASGYQDNIARLEEEIRHLKDEMARHLREYQDLLN
                     VKMALDVEIATYRKLLEGEESRINLPIQTYSALNFRETSPEQRGSEVHTKKTVMIKTI
                     ETRDGEVVSEATQQQHEVL"
     intron          3853..4904
                     /number=1
     exon            4905..4965
                     /number=2
                     /product="desmin"
     intron          4966..5061
                     /number=2
     exon            5062..5157
                     /number=3
                     /product="desmin"
     intron          5158..5305
                     /number=3
.....
.....
.....
     exon            10692..>11463
                     /number=9
                     /product="desmin"
     polyA_signal    11458..11463
```

```
BASE COUNT     2734 a   3160 c   3331 g   2765 t
ORIGIN
        1 ctgcagatgg atggtgggga tagttgtaca acaatgtgaa tgtacttgac gccactgaac
       61 tgtatactaa aaaatggctc aaatggtgaa ttttatttta tgtatatttt atcacaatgt
      121 aaaagtgact cagctgtcaa tgtactaaaa tggaaccatc tctaacgtgt atgtcattga
........
........
........
    11881 ttttttctc tctctctttt tttctttttg agatggagtc ttgctgtgtc acccaggctg
    11941 gagtgcagca gtgtgatcac cgcttactgc agcctggact tcctgagctc
//
```

GenBank ermöglicht den Download von vorhandenen Flatfiles für einen spezifischen Organismus. In einem ersten Vorverarbeitungsschritt wurden aus diesen Dateien alle die herausgefiltert, die im *DEFINITION* Feld “gene”, aber nicht “pseudogene” und im *LOCUS* Feld “DNA” stehen hatten. Weiterhin mußten die Dateien die eigentlichen Sequenzen mitliefern und nicht über zusätzliche Verweise andere Flatfiles referenzieren. Da die Organismen–spezifische *GenBank* Suche offenbar nicht völlig korrekt funktioniert und sich teilweise Sequenzen von anderen Organismen einschleichen, ist es außerdem notwendig, das *ORGANISM* Feld auf Korrektheit zu prüfen. Eventuell vorhandene mitochondriale Sequenzen (oder bei Pflanzen Chloroplastensequenzen) wurden dabei ebenfalls entfernt.

Im zweiten Schritt wurden die *mRNA* sowie die *exon* und *intron* Einträge der Feature Tabelle, benutzt, um die Positionen der Exons und Introns zu bestimmen. Die Einträge *mRNA* und *exon, intron* liefern oftmals redundante Informationen. Der *mRNA* Eintrag wurde verwendet, wenn er keinen Verweis auf andere Flatfiles enthielt. Andernfalls oder bei dessen Fehlen wurde auf einzelne *exon, intron* Einträge zurückgegriffen.

Jedes Exon und Intron muß außerdem eine Mindestlänge erfüllen. Das ist notwendig, da für die Entropieschätzung die letzten 16 Zeichen berücksichtigt werden sollen. Damit kann man die Matchlänge nur für eine Sequenz bestimmen, die mindestens 32 Zeichen lang ist. Da in diesem Fall der Mittelwert der Matchlängen nur aus einer Realisierung besteht, habe ich die Mindestlänge auf 40 Zeichen gesetzt. In den verwendeten Sequenzen gab es nur äußerst wenig Exons und Introns, die diese Bedingung nicht erfüllen konnten.

Da die Entropie von benachbarten Exons und Introns bestimmt werden soll, wurde beim Parsen darauf geachtet, daß es keine Lücken zwischen den Exons und Introns gibt. Solche Lücken können durch unvollständige Annotierung oder zu kurze Länge entstehen. In diesem Fall wurde die kontinuierliche Exon–Intronfolge an der Stelle unterbrochen und nach der Lücke eine neue Folge begonnen.

Das Ergebnis des Parsens ist eine Datei, die in jeder Zeile den *GenBank* Locus, ein *E* oder *I* für Exon bzw. Intron sowie die Exon– oder Intronsequenz enthält. Innerhalb des gleichen Locus Eintrages haben aufeinanderfolgende Zeilen abwechselnd Exon– und Introneinträge. Dieses Format ist gut für die Entropieschätzung der Exons und Introns sowie den Vergleich dieser Werte geeignet.

```
HUMDES E cctcgccgcatccactctccggccggccgcctgcccgccgcctcctccgtgcgcccgccagcctc....
HUMDES I gtgagggcccggcaccccagactcctctttctgcgggcagggcacaggaggctaggcctgggggc....
HUMDES E gctgcaggaggagattcagttgaaggaagaagcagagaacaatttggctgccttccgagcg
HUMDES I gtgagtgcccttcttttcccccttgcatggcctctcgcgtttgctctgccccacctgggtcgcgga....
HUMDES E gacgtggatgcagctactctagctcgcattgacctggagcgcagaattgaatctctcaacgagga....
......
......
```

Um ein möglichst breites Spektrum von Organismen zu betrachten, habe ich die folgenden Spezies ausgewählt: den Fadenwurm (Caenorhabditis elegans), die Fruchtfliege (Drosophila melanogaster), 3 Säugetiere: das Rind (Bos taurus), den Mensch (Homo sapiens), die Maus (Mus musculus) und 3 Pflanzen: die Ackerschmalwand (Arabidopsis thaliana), den Reis (Oryza sativa) und den Mais (Zea mays). Für diese Organismen sind ausreichend bekannte und annotierte Gene vorhanden, so daß mindestens die Datenmenge von Farach et al. erreicht wurde. Tabelle 8.1 zeigt für jede Spezies die Anzahl der Exons (# Exons), deren Durchschnitts– und mediane Länge (AveLen und MedLen) sowie die gleichen Daten für die Introns.

	# Exons	AveLen	MedLen	# Introns	AveLen	MedLen
Arabidopsis	2623	239.41	153	2286	153.74	97
Bos taurus	706	230.17	160	588	649.41	354
Caenorhabditis	905	353.08	181	716	612.82	216
Drosophila	2528	486.50	261	2016	361.05	72
Homo sapiens	16781	220.75	139	15910	740.72	354
Mus musculus	7544	238.47	139	6783	824.62	333
Oryza sativa	1221	272.60	151	1027	340.58	130
Zea mays	1971	219.74	143	1669	228.89	103

Tabelle 8.1: Anzahl und Länge der Exons / Introns für die verwendeten Spezies

Ergebnisse

Zur Schätzung der Entropie aufeinanderfolgendener Exons und Introns habe ich den Match Length Entropieschätzer in C++ implementiert. Die Implementation

unterscheidet sich von dem Pseudokode auf Seite 51 nur darin, daß außerdem die Matchlängenvarianz berechnet wird.

Für jeden Organismus wurde die Entropie und ML–Varianz für alle Exons und Introns bestimmt. Um Vergleichbarkeit mit den Ergebnissen von Farach et al. sicherzustellen, habe ich ebenfalls den Parameter n auf 16 gesetzt. Es wurden folgende Werte ermittelt:

- der Entropiemittelwert aller Exons sowie die Varianz der Entropieschätzung
- der Entropiemittelwert und Varianz der Entropieschätzung aller Introns
- der Mittelwert der ML–Varianz aller Exons sowie dessen Varianz
- der Mittelwert und die Varianz der ML–Varianz aller Introns
- in wieviel Prozent aller Fälle das Exon eine höhere Entropie als das benachbarte Intron hatte
- in wieviel Prozent aller Fälle das Intron eine höhere ML–Varianz als das benachbarte Exon hatte.

Tabelle 8.2 zeigt die Ergebnisse für alle Organismen. Die ersten beiden Spalten zeigen die durchschnittliche Entropie (Ave Ent) und ML–Varianz (Ave ML-Var) für Exons, die Spalten drei bis vier diese Werte für die Introns. Bei allen Spezies haben Exons im Durchschnitt höhere Entropien als Introns. Die Entropieschätzungen sind dabei nahe des maximalen Wertes von 2, bei Arabidopsis ist dieser Wert sogar überschritten. Weiterhin haben Introns im Durchschnitt eine höhere Matchlängenvarianz als Exons.

Abbildung 8.1 zeigt die Verteilung der Entropien für Exons und Introns. Beide Verteilungen überlappen sich stark und sind etwa normalverteilt. Die beiden Kurven wurden ermittelt, indem jeder Entropiewert einem Intervall der Breite 0.05 zugeordnet wurde. Nach der Berechnung aller Differenzen wurde die Anzahl der Werte eines Intervalls durch die Gesamtzahl geteilt, was die relative Häufigkeit dafür angibt, daß ein Entropiewert in diesem Intervall liegt.

Um die Mittelwertdifferenzen der Entropie und Matchlängenvarianz auf statistische Signifikanz zu untersuchen, kann der T–Test angewendet werden. Dafür müssen die Zufallsgrößen normalverteilt sein, was die Kurven in Abbildung 8.1 erfüllen. Weiterhin müssen die Zufallsgrößen unabhängig sein. Dies ist biologisch vertretbar, da ein Exon außer einer konservierten Spleisstelle keinen Bezug zum Nachbarintron haben sollte. Die Unabhängigkeit konnte ein Punktplot bestätigen,

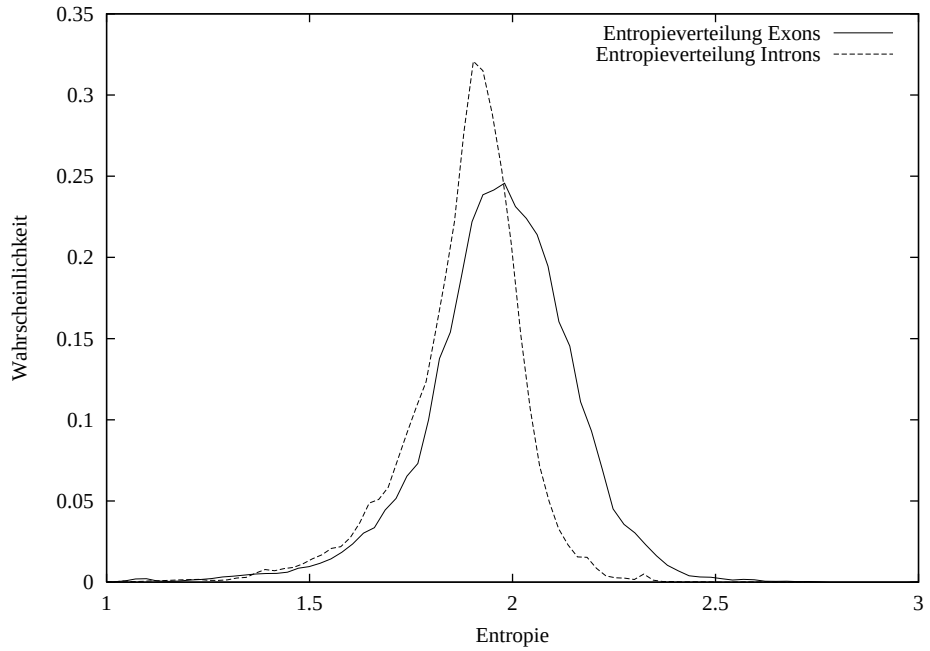


Abbildung 8.1: Verteilung der Exon– und Intronentropie

der die paarweisen Entropien von allen Exons zu ihren Nachbarintrons enthielt und keine Korrelation zeigte. Ein Plot der paarweisen Entropien von allen Exons zu allen Introns eines Gens zeigte im Vergleich ebenfalls keine Korrelation. Da die Varianz der Zufallsgrößen nicht bekannt ist, sondern aus den Daten geschätzt wird, muß man auf die T–Verteilung zurückgreifen, die sich bei diesen hohen Fallzahlen aber nicht von der Normalverteilung unterscheiden wird.

Die letzten beiden Spalten in Tabelle 8.2 geben den p–Wert der Entropie– bzw. ML–Varianzdifferenz an. Der p–Wert sagt dabei aus, mit welcher Wahrscheinlichkeit sich die beobachtete Differenz oder eine noch höhere Differenz ergibt, wenn beide Verteilungen den selben Erwartungswert haben. Ein p–Wert von 0 in Tabelle 8.2 bedeutet, daß der p–Wert kleiner als 1.0E-50 ist. Die beobachteten Differenzen sind zwar klein, aber statistisch hoch signifikant. Eine Ausnahme ist nur der Reis, bei dem es offenbar keinen signifikanten Unterschied der ML–Varianz gibt. Ob dies auf einen ausgewählten Datensatz zurückzuführen ist, oder ob der Reis wirklich eine Ausnahme darstellt, für die es eventuell biologische Ursachen gibt, kann nicht gesagt werden.

	Ave Ent Exon	Ave MLVar Exon	Ave Ent Intron	Ave MLVar Intron	p–Wert Entropie-differenz	p–Wert MLVarianz-differenz
Arabidopsis	2.025	1.024	1.885	1.308	0.00	0.00
Bos taurus	1.973	1.085	1.885	1.431	3.28E-17	7.58E-13
Caenorhabditis	1.991	1.080	1.820	1.615	0.00	2.47E-36
Drosophila	1.997	1.123	1.893	1.404	0.00	1.66E-23
Homo sapiens	1.961	1.153	1.886	1.447	0.00	0.00
Mus musculus	1.986	1.110	1.890	1.804	0.00	0.00
Oryza sativa	1.972	1.215	1.920	1.251	1.38E-07	0.455396
Zea mays	1.979	1.070	1.913	1.381	1.87E-18	5.91E-16

Tabelle 8.2: durchschnittliche Entropie (Ave Ent) und Matchlängenvarianz (Ave MLVar) für Exons und Introns sowie der p–Wert für die Signifikanz der Differenzen

In Tabelle 8.3 ist in den ersten beiden Spalten die Prozentzahl der Exons, die eine höhere Entropie als das Nachbarintron, bzw. die Prozentzahl der Introns, die eine höhere ML–Varianz, als das benachbarte Exon haben, dargestellt. Die Ergebnisse decken sich mit denen aus Tabelle 8.2. Exons haben im Vergleich zu den benachbarten Introns eine höhere Entropie. Gleiches gilt für Introns und die Matchlängenvarianz.

Auch diese Ergebnisse sind für alle Spezies statistisch signifikant, was der Mann–Whitney–U Test für unabhängige Paardifferenzen zeigt. Die p–Werte sind alle kleiner als 0.001.

Die Ergebnisse von Farach et al. können damit bestätigt werden und lassen sich auch auf andere Spezies verallgemeinern. Allerdings unterscheiden sich die Ergebnisse des größeren Datensatzes etwas von denen aus [21]. Menschliche Exons haben zu 67 % eine höhere Entropie, Introns zu 71 % eine höhere ML–Varianz, was niedriger ist als die von Farach et al. erwähnten 73 bzw. 80 %. Dies könnte bedeuten, daß Farach's Datensatz nicht ganz repräsentativ für das menschliche Genom ist. So sind mittlerweile sicherlich viele menschliche Gene gefunden worden, deren Charakteristiken von der Mehrheit der Gene abweichen und daher als untypisch gelten können. Das könnte erklären, warum die Prozentzahlen nicht ganz so hoch sind.

Ich möchte aber betonen, daß auch der hier verwendete Datensatz nicht repräsentativ für das ganze Genom sein muß. So sind heute zwar viele menschli-

	% Exons höhere Entropie	% Introns höhere MLVarianz
Arabidopsis	73.647	66.835
Bos taurus	68.451	74.085
Caenorhabditis	76.664	78.454
Drosophila	65.740	61.647
Homo sapiens	67.273	70.935
Mus musculus	67.561	72.083
Oryza sativa	64.753	64.602
Zea mays	62.050	68.517

Tabelle 8.3: Prozentzahl der Exons mit höherer Entropie und Prozentzahl der Introns mit höherer ML–Varianz

che Gene bekannt, allerdings ist nur ein Teil davon experimentell bestätigt. Die restlichen Gene sind hypothetisch und wurden u.a. mit Hilfe von Genvorhersage Programmen (siehe Kap. 8.4.1) gefunden. Diese Programme finden Gene in einer nicht annotierten Sequenz mit Hilfe verschiedener Heuristiken und beschleunigen die experimentelle Bestätigung erheblich. Es ist daher nicht auszuschließen, daß einige Gene, die den Heuristiken nicht genügen, noch nicht entdeckt wurden. Diese Gene könnten unterschiedliche Charakteristiken aufweisen, auch in Bezug auf die Entropie ihrer Exons und Introns.

8.3 Berücksichtigung des Differenzenbetrags

Im letzten Abschnitt wurde gezeigt, daß bei allen untersuchten Organismen etwa zwei Drittel der Exons eine höhere Entropie als ihr Nachbarintron und etwa ebenso viele Introns eine höhere ML–Varianz als das Nachbarexon haben. Es ist interessant zu prüfen, wie sich diese Prozentzahlen verhalten, wenn man zusätzlich den Betrag der Differenz mit berücksichtigt.

Dazu wird in Abbildung 8.2 zunächst die Verteilung der Differenzen für menschliche Exons gezeigt, die wieder durch Zuordnung zu Intervallen der Breite 0.05 im Wertebereich -7 bis +4 erstellt wurde. Da immer die Differenz der Entropie und Matchlängenvarianz eines Exons zum Nachbarintron berechnet wurde, befindet sich das Maximum der Entropiedifferenzverteilung im positiven, das Maximum der Verteilung der ML–Varianzdifferenzen im negativen Bereich. Die

positiven Werte an den Randbereichen zeigen, daß es ML–Varianzen gibt, die eine größere Differenz als -7 bzw. +4 haben. Weiterhin haben die Entropiedifferenzen eine deutlich geringere Streuung als die Differenzen der ML–Varianz.

Abbildung 8.2 zeigt, daß es nicht sinnvoll ist, höhere Differenzen als 0.5 zu untersuchen, da diese Differenzbeträge nur sehr selten vorkommen und dadurch keine gesicherte Aussage mehr zulassen.

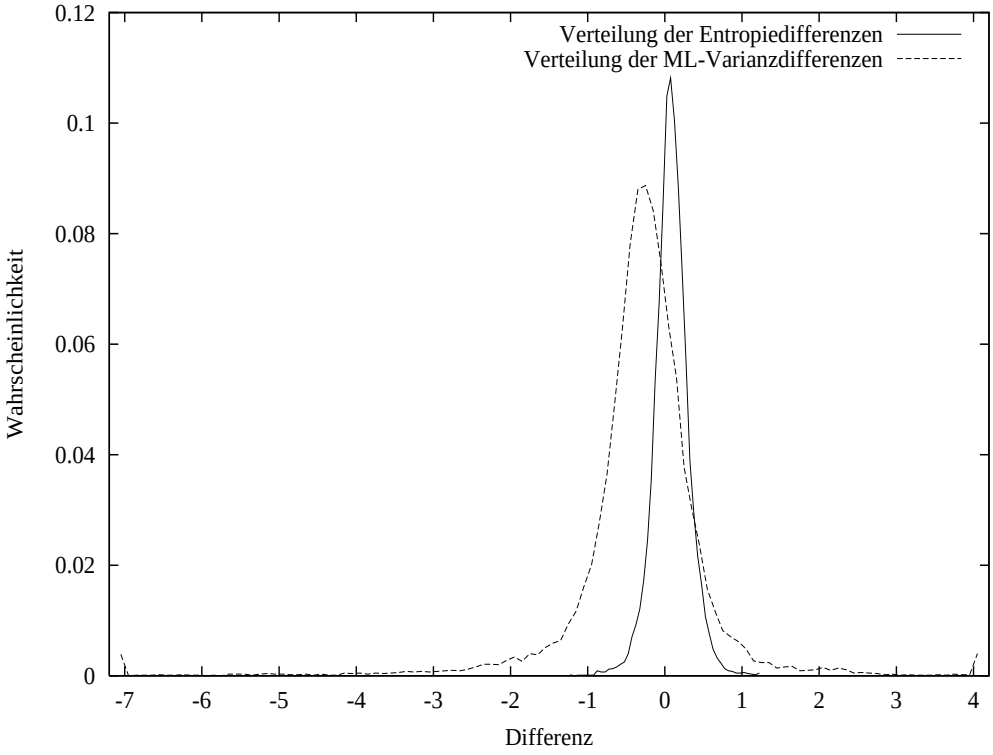


Abbildung 8.2: Verteilungen der Entropie– und ML–Varianzdifferenzen zwischen menschlichen Exons und ihren Nachbarintrons

Wenn der Betrag der Entropiedifferenz zwischen einem menschlichen Exon und seinem Nachbarintron größer als 0 ist, dann hat zu 67 % das Exon die höhere Entropie. Dies ist das Ergebnis des vorherigen Abschnitts. Die Frage ist: Zu wieviel Prozent hat das menschliche Exon die höhere Entropie, wenn der Betrag der Entropiedifferenz größer als beispielsweise 0.25 ist?

Da Exons eine durchschnittlich höhere Entropie haben, würde man erwarten, daß diese Prozentzahl mit dem Betrag der Entropiedifferenz steigt. Die Hoffnung ist, damit einen starken Diskriminator zu finden, der Exons und Introns unterscheiden kann. In der Tat konnte dies beobachtet werden.

Um diese Frage zu untersuchen, habe ich für alle Exon–Intron Paare den Betrag der Entropiedifferenz berechnet. Dieser Wert wurde wieder einem Intervall der Breite 0.05 zugeordnet, wobei sich der Wertebereich von 0 bis 0.5 erstreckte. Dann wurde für jedes Intervall die relative Häufigkeit ermittelt, mit der das Exon eine um diesen Betrag höhere Entropie hatte. In der Implementierung besaß dazu jedes Intervall zwei Zähler, die die Anzahl der Exons bzw. Introns mit der höheren Entropie speicherten.

Die Ergebnisse dieser Untersuchung sind in Abbildung 8.3 gezeigt. Die Punkte repräsentieren jeweils den Mittelpunkt eines Intervalls. Der letzte Punkt entspricht dem Intervall $(0.5, \infty)$. Damit gibt jeder Punkt an, mit welcher Prozentzahl das Exon die höhere Entropie hatte, falls der Betrag der Entropiedifferenz in dem Intervall liegt, dessen Mitte durch den Punkt repräsentiert wird.

Für alle verwendeten Organismen ergaben sich ähnliche Kurven. In Abbildung 8.3 werden der Übersicht wegen nur vier zufällig gewählte Spezies (Arabidopsis, Homo sapiens, Mus musculus, Caenorhabditis) gezeigt. Es ist ein linearer Aufwärtstrend zu erkennen, der die obige Erwartung bestätigt. Weshalb die Kurven nicht ganz linear sind und teilweise im letzten Intervall wieder fallen, kann allerdings nicht erklärt werden.

In Analogie zur Entropie sollte etwas ähnliches auch für die Matchlängenvarianzen gelten. Da hier die Introns den höheren durchschnittlichen Wert haben, sollte sich die Exon–Prozentzahl jetzt aber entgegengesetzt verhalten. Je größer der Betrag ist, desto geringer sollte die Prozentzahl der Exons sein, die eine um diesen Betrag höhere ML–Varianz als das Nachbarintron haben.

Auch diese Erwartung bestätigte sich, wie Abbildung 8.4 zeigt. Da sich bei allen Organismen ähnliche Kurven ergaben, werden auch hier nur vier Spezies gezeigt. Da die Matchlängenvarianz stärker als die Entropie streut, sind die Schwankungen dieser Kurven recht stark, was besonders bei Arabidopsis und Caenorhabditis deutlich wird. Auch hier kann ich keine Erklärung für diese Schwankungen geben. Eventuell ist dies auf eine zu geringe Fallzahl zurückzuführen, da die Kurven für den Mensch und die Maus wesentlich glatter verlaufen. Trotzdem ist meiner Meinung nach auch hier ein linearer Abwärtstrend sichtbar.

Zusammenfassend läßt sich sagen, daß ein starker Diskriminator die Höhe der Entropie– und ML–Varianzdifferenzen berücksichtigen sollte. So beträgt die Wahrscheinlichkeit, daß ein menschliches Exon die höhere Entropie hat, wenn

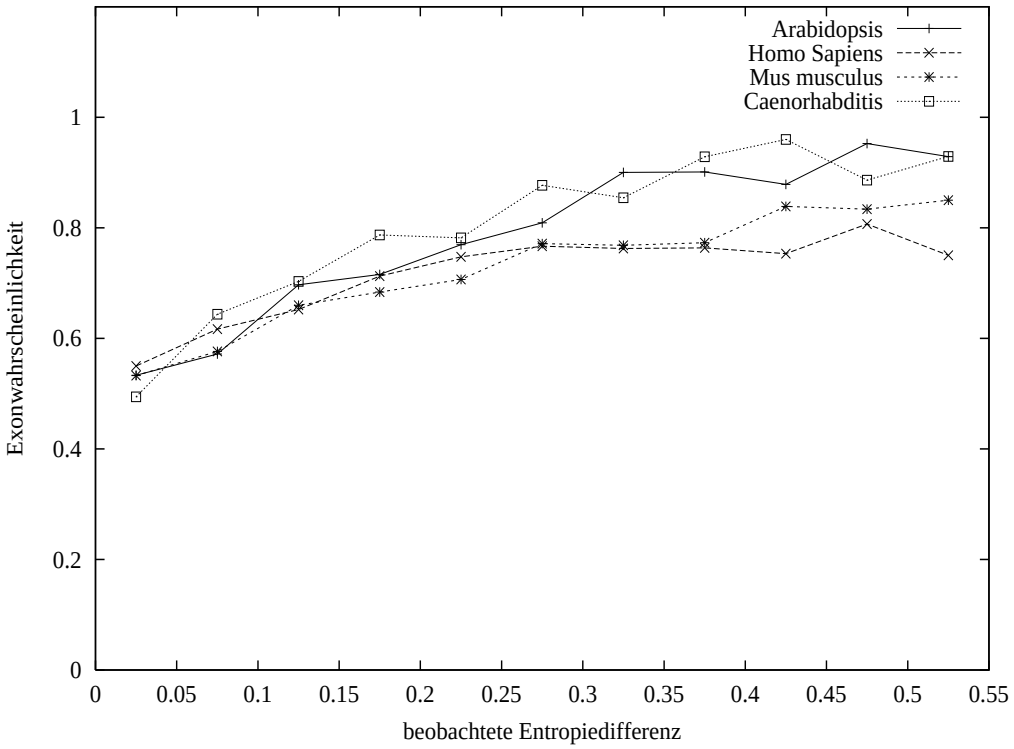


Abbildung 8.3: Exonwahrscheinlichkeit bei beobachteter Entropiedifferenz

eine Differenz von 0.5 beobachtet wurde, etwa 80 %. Analog dazu beträgt die Wahrscheinlichkeit, daß ein menschliches Intron die höhere ML–Varianz hat, wenn eine Differenz von 0.5 beobachtet wurde, auch etwa 80 %.

8.4 Verwendung bei der Vorhersage von Exons

In diesem Abschnitt soll untersucht werden, ob der Entropieunterschied von Exons und Introns sowie die Berücksichtigung der Differenz bei der Vorhersage von Exons benutzt werden kann. Dazu werden zuerst einige Bemerkungen über Genvorhersage Programme gemacht.

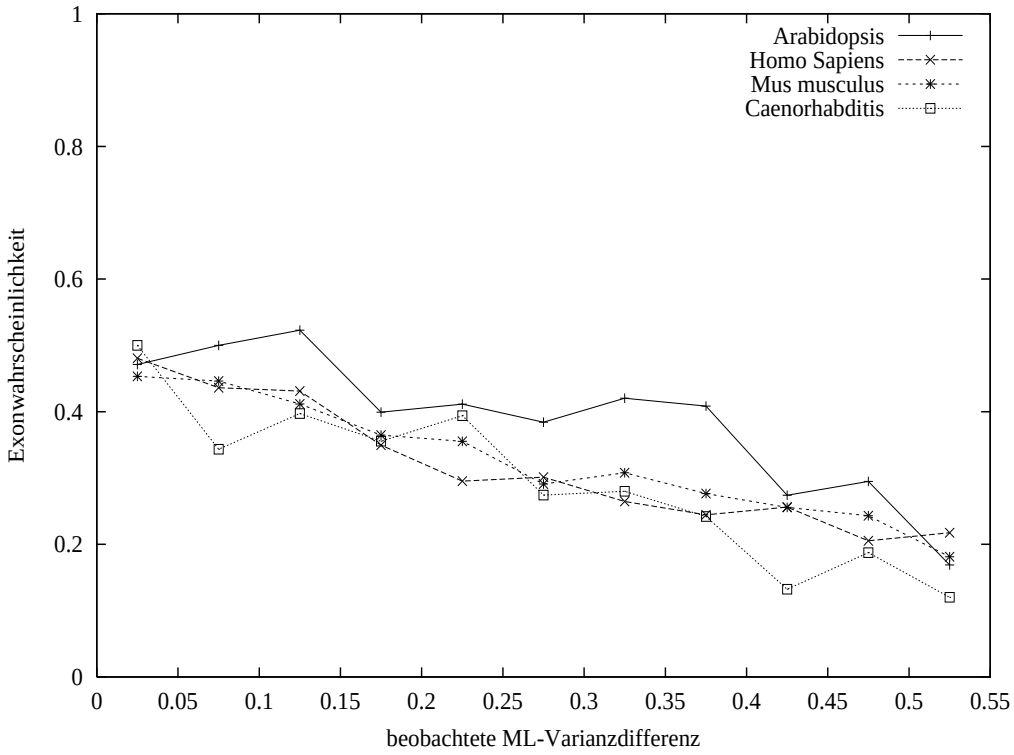


Abbildung 8.4: Exonwahrscheinlichkeit bei beobachteter ML–Varianzdifferenz

8.4.1 Genvorhersage Programme

Die *Vorhersage von Genen* in einer nicht annotierten Sequenz ist eine der großen Herausforderungen der Bioinformatik. Seit Jahren wird sich weltweit intensiv mit diesem Thema beschäftigt. Für eine detaillierte Einführung in diese Thematik verweise ich auf die exzellenten Reviews von Claverie [13], Burge und Karlin [10] und Haussler [27]. Einen Vergleich der wirklichen Leistungsfähigkeit von Genvorhersage Programmen haben Burset und Guigo [11] durchgeführt.

Die schnell fortschreitende Sequenzierung von ganzen Genomen (darunter das menschliche) liefert eine riesige Menge an Sequenzen. Allerdings sind nur von einem Bruchteil dieser Sequenzen die enthaltenen Gene bekannt, da das Finden von Genen immer noch eine Menge Laborarbeit erfordert. Es ist daher wünschens-

wert, Gene nur anhand der Sequenzinformation zu finden. Dies ist die Aufgabe von *Genvorhersage Programmen*.

Für das Finden von prokaryotischen Genen ist es erfolgversprechend, nach langen *Open Reading Frames* (ORF) zu suchen. Ein ORF ist eine lange Folge von Codons, die kein Stop Codon enthält. Ein genügend langes ORF korrespondiert fast immer zu einem Gen.

Bei Eukaryoten ist diese Methode nicht einsetzbar. So können Introns Stop Codons enthalten, was die Codonabfolge der Exons nicht beeinträchtigt. Genvorhersage bei Eukaryoten erfordert also nicht nur das Finden der Position eines Gens in der gegebenen Sequenz, sondern auch die Einteilung in Exons und Introns.

Die meisten Verfahren zur Genvorhersage verwenden verschiedene Heuristiken. So gibt es an Spleisstellen konservierte Sequenzen, mit denen man einen Teil der wirklichen Spleisstellen entdecken kann. Weiterhin gibt es Unterschiede in der Codonverteilung zwischen Exons und Introns. Die Verteilung von Hexameren (also zwei Codons) erweist sich als noch unterschiedlicher und wird daher oft als Diskriminator benutzt. Die meisten Programme setzen außerdem eine Mindest– und Höchstanzahl von Introns voraus und haben gewisse Vorgaben für die maximale und minimale Länge der Exons und Introns.

Zur Integration dieser Heuristiken in eine Vorhersage eines kompletten Gens, verwendet man unterschiedliche Ansätze. Sehr populär sind *Hidden Markov Modelle* auf denen *GENSCAN* [9] und *GENMARK* [6] basieren. Andere Programme verwenden regelbasierte Systeme (*GeneID* [25]), neuronale Netze (*GRAIL II* [62]), lineare oder quadratische Diskriminatenanalyse (*HEXON* [56], *FGENEH* [57], *MZEF* [63]) oder Entscheidungsbäume (*MORGAN* [52]).

8.4.2 Bewertung von vorhergesagten Exons

Im Folgenden soll die Ausgabe eines Genvorhersage Programms bewertet werden. Die Hoffnung dabei ist, daß einerseits korrekte Exons durch einen Entropieunterschied bestätigt werden können, andererseits einige falsche Exons durch eine niedrigere Entropie im Vergleich zum Intron identifiziert werden können.

Für das praktische Experiment habe ich das Programm *MZEF* [63] von Michael Zhang ausgewählt. *MZEF* (Michael Zhang's Exon Finder) versucht in einer Sequenz alle internen Exons (alle Exons außer das erste und letzte) zu finden. Das Programm verwendet Spleisstellenvorhersage und quadratische Diskriminatenanalyse. Dabei wird der mehrdimensionale Merkmalsraum, der Exons und nichtkodierende Abschnitte einer Trainingsmenge enthält, durch eine quadratische Diskriminatenfunktion in zwei Bereiche so aufgeteilt, daß jeder Bereich

möglichst homogen aus Exons bzw. nichtkodierenden Abschnitten besteht. Ich habe *MZEF* gewählt, weil dieses Programm relativ gute Vorhersagen generiert, die Ausgabe leicht weiterverarbeitet werden kann und das Programm nur von einer kleinen Anzahl an Parametern abhängt. Eine ausführbare Version für das Linux Betriebssystem findet sich im Internet unter [29].

Für das Experiment wurde ein Datensatz gewählt, der vollständig annotierte menschliche Gene enthält und unter [3] heruntergeladen werden kann. Aus diesem Datensatz wurden alle Gene ausgeschlossen, die weniger als drei Exons enthielten, da *MZEF* nur interne Exons vorhersagt. Weiterhin wurden die Gene ausgeschlossen, die in dem Datensatz aus Kapitel 8.2 enthalten waren. Das Ergebnis waren 261 Gene, die 1422 interne Exons enthielten.

Für diese 261 Sequenzen berechnete *MZEF* eine Vorhersage der internen Exons, wobei die Standardparameter verwendet wurden. Insgesamt wurden 1447 Exons von *MZEF* vorhergesagt. Ein Exonprediktor kann folgende Fehler machen [11]. Ein irrtümlich vorgesagtes Exon in einer Intronsequenz wird als falsches Exon bezeichnet. Ein fehlendes Exon ist ein Exon, daß überhaupt nicht erkannt wurde. Weiterhin kann ein Exon korrekt erkannt worden sein, aber die genauen Grenzen wurden nicht exakt bestimmt. Abbildung 8.5 stellt diese Fehler graphisch dar.

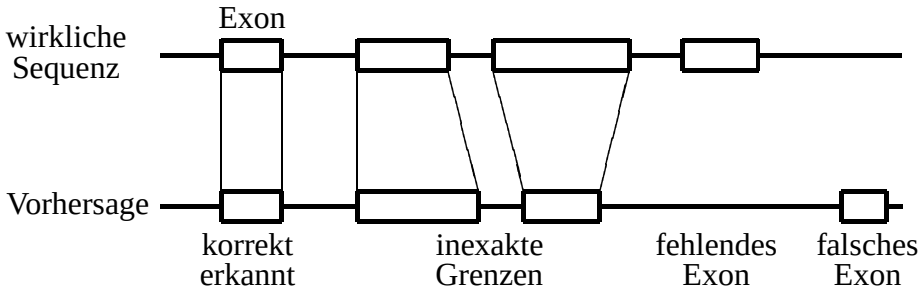


Abbildung 8.5: Fehler eines Genvorhersage Programms auf der Exonebene

Da die Entropie als Maß nicht von einzelnen Positionen abhängt, sondern für Sequenzbereiche berechnet wird, erschien mir folgende Klassifikation der vorhergesagten Exons als vernünftig. Jedes vorhergesagte Exon wurde in eine der drei Klassen eingeteilt:

- **Korrekte Exons (K):** Die beiden Spleisstellen stimmen exakt überein.

- **Teilweise korrekte Exons (TK):** Jede der beiden vorhergesagten Spleisstellen ist nicht mehr als N Positionen von der korrekten Spleisstelle entfernt.

- **Nicht korrekte Exons (NK):** Mindestens eine der beiden vorhergesagten Spleisstellen ist mehr als N Positionen von der korrekten Spleisstelle entfernt.

Teilweise korrekte Exons besitzen also höchstens $2 * N$ Intron–Zeichen. Im Folgenden wurden für N die Werte 25 und 50 verwendet. Abbildung 8.6 zeigt diese Klassifikation noch einmal graphisch.

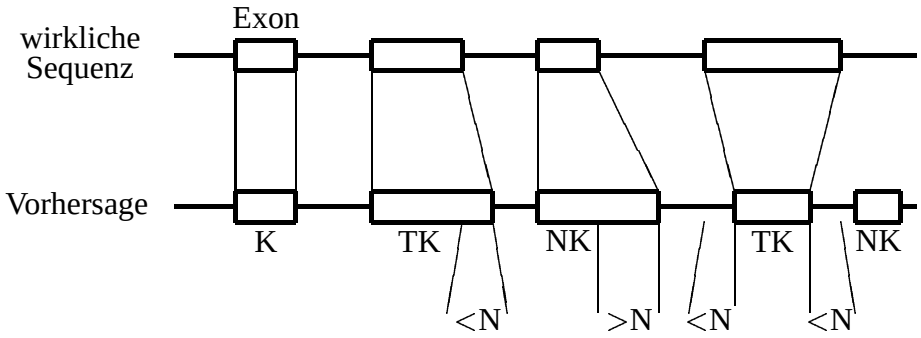


Abbildung 8.6: Schema der Klassifikation in korrekte, teilweise korrekte und nicht korrekte Exons

Anschließend wurden aus jeder Gensequenz die von *MZEF* vorhergesagten Exon– und Intronsequenzen extrahiert und für alle benachbarten Exon–Intronpaare die Entropie und Matchlängenvarianz berechnet. Mit der Entropie– und ML–Varianzdifferenz wird jedes vorhergesagte Exon in eine der folgenden drei Klassen eingeteilt. Dabei sei k der Betrag einer Entropie– bzw. ML–Varianzdifferenz.

- **Bestätigung:** Die Entropie dieses Exons ist um mindestens k größer als die Entropie des Nachbarintrons, und die ML–Varianz des Exons ist um mindestens k niedriger als die des Nachbarintrons.

- **Ablehnung:** Die Entropie dieses Exons ist um mindestens k niedriger und die ML–Varianz ist um mindestens k höher als die Werte für das benachbarte Intron.

- **Keine Angabe:** sonst

Der Parameter k wurde von 0 bis 0.6 in Schritten von 0.05 variiert.

Von den 1447 vorhergesagten Exons waren 919 korrekt, was eine Spezifität von 63.5 % ergibt. Das dieser Wert niedriger ist, als die von Zhang in [63] angegebenen 86 %, zeigt, daß die Performance von Genvorhersage Programmen wesentlich vom verwendeten Testdatensatz abhängt [11]. Es sei bemerkt, daß auf dem Gebiet der Genvorhersage die Spezifität generell als Anteil der korrekten an allen vorhergesagten Exons definiert ist [11].

Ich möchte an dieser Stelle betonen, daß es bei dieser Untersuchung nicht darum geht, die Genauigkeit der Vorhersagen von *MZEF* zu bewerten. Das ist schwer möglich, da nicht eingeschätzt werden kann, in wie weit sich der vorliegende Datensatz mit dem Trainingsdatensatz von *MZEF* überlappt. Hier soll nur untersucht werden, wieviel der bestätigten Exons wirklich (teilweise) korrekt sind und wieviel der als falsch klassifizierten Exons nicht korrekt sind. Dieser Sachverhalt wird in Tabelle 8.4 noch einmal dargestellt. Die Hoffnung ist, daß sich viele korrekte und teilweise korrekte Exons in der ersten Tabellenzeile finden und nur wenige in der letzten Zeile.

	korrekt (K)	teilweise korrekt (TK)	nicht korrekt (NK)
bestätigt			
keine Angabe			
abgelehnt			

Tabelle 8.4: Schema der Klassifikation der vorhergesagten Exons

Zur Berechnung der Entropie und ML–Varianz wurde wieder der Match Length Entropieschätzer mit Parameter $n = 16$ verwendet.

Die Ergebnisse für $N = 25$ und $N = 50$ sind in Tabelle 8.5 und 8.6 gezeigt. Die erste Spalte gibt den Parameter k, also die Mindestdifferenz, an. Die nächsten drei Spalten zeigen wieviel der bestätigten Exons korrekt, teilweise korrekt und nicht korrekt waren. In Spalte fünf und sechs ist der Anteil der korrekten bzw. korrekten oder teilweise korrekten Exons an allen Exons, die bestätigt wurden, dargestellt. Die letzte Spalte ist der Anteil der bestätigten Exons von der Gesamtheit der vorhergesagten 1447 Exons (als *All* bezeichnet).

Diff	K	TK	NK	$\frac{K}{K+TK+NK}$	$\frac{K+TK}{K+TK+NK}$	$\frac{K+TK+NK}{All}$
0.00	685	89	254	0.67	0.75	0.71
0.05	607	78	207	0.68	0.77	0.62
0.15	449	48	126	0.72	0.80	0.43
0.20	369	37	92	0.74	0.82	0.34
0.25	291	26	60	0.77	0.84	0.26
0.30	219	19	40	0.79	0.86	0.19
0.35	168	15	31	0.79	0.86	0.15
0.40	127	14	23	0.77	0.86	0.11
0.45	93	11	16	0.78	0.87	0.08
0.50	71	10	8	0.80	0.91	0.06
0.55	55	8	4	0.82	0.94	0.05
0.60	45	7	2	0.83	0.96	0.04

Tabelle 8.5: Ergebnisse für bestätigte Exons und $N = 25$

Diff	K	TK	NK	$\frac{K}{K+TK+NK}$	$\frac{K+TK}{K+TK+NK}$	$\frac{K+TK+NK}{All}$
0.00	685	129	214	0.67	0.79	0.71
0.05	607	109	176	0.68	0.80	0.62
0.15	449	67	107	0.72	0.83	0.43
0.20	369	49	80	0.74	0.84	0.34
0.25	291	33	53	0.77	0.86	0.26
0.30	219	24	35	0.79	0.87	0.19
0.35	168	19	27	0.79	0.87	0.15
0.40	127	17	20	0.77	0.88	0.11
0.45	93	13	14	0.78	0.88	0.08
0.50	71	11	7	0.80	0.92	0.06
0.55	55	9	3	0.82	0.96	0.05
0.60	45	8	1	0.83	0.98	0.04

Tabelle 8.6: Ergebnisse für bestätigte Exons und $N = 50$

Es ist deutlich zu erkennen, daß bei steigender Differenz auch der Anteil der korrekten an den bestätigten Exons steigt. Dasselbe gilt für den Anteil der korrekten und teilweise korrekten. In gleichem Maße sinkt aber auch der Anteil der bestätigten Exons. Die Spezifität liegt aber generell höher als die 63.5 %, die *MZEF* für diesen Datensatz erreichte. So konnte beispielsweise aus 43 % von allen vorhergesagten Exons 72 % korrekte Exons herausgefiltert werden, wenn die Differenz d 0.15 beträgt.

Der Unterschied zwischen $N = 25$ und $N = 50$ ist, daß im zweiten Fall mehr teilweise korrekte Exons bestätigt werden, was darauf hindeutet, daß einige der nicht korrekten Exons offenbar nicht völlig falsch sind, sondern gemeinsame Bereiche mit dem wahren Exon teilen.

Gewöhnlich haben neu sequenzierte und noch nicht annotierte Sequenzen eine niedrigere Gendichte als die Sequenzen der Trainings– und Testdatensätze. Da Genvorhersage Programme die Eigenschaft haben, bei Gen–armen Sequenzen viele falsche Exons vorherzusagen, könnte die hier vorgestellte Methode dazu beitragen, aus der großen Masse an Vorhersagen eine kleine Menge herauszufiltern, die mit hoher Wahrscheinlichkeit viele korrekte oder teilweise korrekte Exons enthält.

Es wäre außerdem hilfreich, wenn aufgrund der Entropiedifferenzen einige nicht korrekte Exons mit hoher Spezifität erkannt werden könnten. Allerdings versagte die Methode bei dieser Aufgabenstellung. So waren weitgehend unabhängig von der Differenz d nur etwa 50 % der abgelehnten Exons wirklich nicht korrekt. In vielen Fällen lag dieser Anteil sogar unter 50 %.

8.5 Zusammenfassung

In diesem Kapitel wurden die unterschiedlichen Entropien von Exons und Introns untersucht. Es wurde gezeigt, daß nicht nur für den Mensch, sondern für alle ausgewählten Spezies Exons im Durchschnitt eine signifikant höhere Entropie haben, was den Erwartungen widerspricht. Über die Gründe dafür können allerdings nur Vermutungen geäußert werden. Weiterhin haben Exons in etwa zwei Drittel der Fälle eine höhere Entropie als das Nachbarintron. Auch diese Ergebnisse sind für alle untersuchten Organismen statistisch signifikant.

Für die Matchlängenvarianz ergibt sich auch ein Unterschied. Konform mit der Erwartung haben hier Introns einen durchschnittlich höheren Wert, was für die meisten Organismen ebenfalls signifikant ist. Im Vergleich zum benachbarten Exon haben die Introns in rund 70 % der Fälle die höhere ML–Varianz.

Wenn man den Betrag der Entropie– und ML–Varianzdifferenzen mit berücksichtigt, erhält man einen guten Diskriminator. Es wurde beobachtet, daß je höher dieser Betrag ist, desto wahrscheinlicher hat das Exon die höhere Entropie. Dasselbe gilt für die Matchlängenvarianz, wobei hier mit steigendem Betrag das Intron den höheren Wert hat.

Weiterhin wurde gezeigt, daß unterschiedliche Entropie und ML–Varianzdifferenzen der Exons und Introns verwendet werden können, um aus allen Vorhersagen eines Exonprediktors eine Menge an bestätigten Exons zu extrahieren, die nur wenige falsche Vorhersagen enthält. Die Spezifität kann über die Höhe der geforderten Differenz d eingestellt werden und bestimmt auch, wie groß die resultierende Menge ist. Leider gelingt es nicht, mit hoher Genauigkeit falsche Exons als solche zu klassifizieren. Außerdem ist diese Methode nicht geeignet, um fehlende Exons zu entdecken und damit die Sensitivität zu verbessern.

Es wäre sehr interessant zu untersuchen, wie sich die unterschiedlichen Eigenschaften der Exons und Introns in Bezug auf Entropie und ML–Varianz schon beim Erstellen der Exonvorhersage verwenden lassen. Da die Genvorhersage Programme oft auf der Kombination von Heuristiken basieren, könnte die hier untersuchte Heuristik mit integriert werden, was eventuell zu einer höheren Spezifität führt. Mir ist zum momentanen Zeitpunkt nicht bekannt, daß ein publiziertes Genvorhersage Programm diese Entropieunterschiede mit benutzt.

Kapitel 9

Charakterisierung von Spleisstellen

Spleissen ist ein hochspezifischer Vorgang bei der Genexpression in Eukaryoten. Die genauen Grenzen des Introns müssen dabei erkannt und anschließend zwei Exons zusammengefügt werden. Daher muß es bestimmte Sequenzen um diese Spleisstellen geben, die eine eindeutige Erkennung ermöglichen. In diesem Kapitel soll mit Hilfe der *bedingten Entropie* untersucht werden, welche Positionen der Spleisstellen für den Erkennungsvorgang wichtig sind. Dafür wird zuerst die bedingte Entropie erklärt und einige Aussagen über Spleisstellen gemacht. Anschließend wird für verschiedene Organismen der Informationsgehalt der Positionen um die Spleisstellen untersucht.

9.1 Bedingte Entropie

Die in Kapitel 3.1 eingeführte Entropie war definiert als

$$H(P) = -\sum_{i=1}^{n} p(x_i)\ log_2(p(x_i)),$$

und mißt die Zufälligkeit und Unsicherheit einer Quelle.

Die *bedingte Entropie* ist analog zur Entropie definiert, verwendet aber bedingte Wahrscheinlichkeiten.

Definition 9.1 (bedingte Entropie) *Seien zwei Quellen* $A = ((x_1, \ldots, x_n), (p(x_1), \ldots, p(x_n)))$ *und* $B = ((y_1, \ldots, y_m), (p'(y_1), \ldots, p'(y_m)))$ *gegeben. Dann ist die bedingte Entropie von* A*, gegeben daß* B *das Zeichen* y_i *generiert hat, definiert als*

$$H(A|B = y_i) = -\sum_{j=1}^{n} p(x_j|y_i)\ log_2(p(x_j|y_i)),$$

wobei $p(x_j|y_i)$ *die Wahrscheinlichkeit ist, daß* A *das Zeichen* x_j *ausgibt, wenn bekannt ist, daß* B *das Zeichen* y_i *generiert hat.*
Die bedingte Entropie von A *gegeben* B *ist definiert als*

$$H(A|B) = -\sum_{i=1}^{m} p'(y_i) \sum_{j=1}^{n} p(x_j|y_i)\ log_2(p(x_j|y_i)).$$

Damit gibt $H(A|B = y_i)$ die Unsicherheit über das nächste Zeichen von A an, wenn B y_i ausgegeben hat. $H(A|B)$ gibt dann die verbleibende Unsicherheit der Quelle A an, wenn die Ausgabe von B bekannt ist.

Es gilt: $H(A|B) \leq H(A)$. Wenn $H(A|B) < H(A)$, dann beseitigt B einen Teil der Unsicherheit über A. Die Gleichheit gilt nur bei Unabhängigkeit von A und B. In diesem Fall liefert B keine Information über das nächste Zeichen von A. Wenn $H(A|B) = 0$, ist das folgende Zeichen von A vollständig durch die Ausgabe von B bestimmt.

Bsp.: Sei B eine Quelle mit $((a, b), (p(a) = 0.5, p(b) = 0.5))$. A sei eine von B abhängige Quelle mit den bedingten Wahrscheinlichkeiten $p(A = a|B = a) = 0.8$, $p(A = b|B = a) = 0.2$, $p(A = b|B = a) = 0.25$, $p(A = b|B = b) = 0.75$. Dann ist $H(A|B = a) = 0.72$ und $H(A|B = b) = 0.81$. Die Unsicherheit über A, wenn man B kennt, ist $H(A|B) = 0.765$. Obwohl B völlig zufällig Zeichen ausgibt, ist die bedingte Entropie von A kleiner, da A häufig das selbe Zeichen generiert wie B. Dadurch verringert sich die Zufälligkeit von A, wenn B vorher bekannt ist.

9.2 Spleisstellen

Jedes Intron hat zwei Spleisstellen, eine am 5'– und eine am 3' – Ende. Dabei bezeichnet man den Intronanfang (5' – Ende) als *Donorstelle*, das Intronende (3' – Ende) als *Acceptor* (siehe Abb. 9.1).

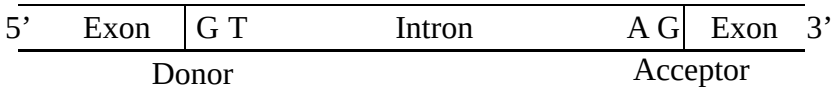


Abbildung 9.1: Schema von typischen Spleisstellen

An den Enden der Introns gibt es konservierte Sequenzen, die eine Erkennung ermöglichen. So beginnt ein Intron in den meisten Fällen mit einem *GT Dinukleotid* und endet mit *AG*. Weiterhin befindet sich am Ende des Introns meist ein pyrimidinreicher (Cytosin und Thymin) Abschnitt.

Allerdings reicht die Information, die in einem einzelnen GT oder AG steckt, nicht aus, um innerhalb eines transkribierten Gens alle Donor– und Acceptorstellen eindeutig zu identifizieren, da GT und AG Dinukleotide auch in Exons vorkommen. Es wäre fatal, wenn der Spleiss–Mechanismus irrtümlich Spleisstellen innerhalb von Exons erkennen und das Exon an dieser Stelle spalten würde.

Um ein beliebiges GT oder AG von einem Donor GT bzw. Acceptor AG zu unterscheiden, benötigt man weitere Informationen, die die benachbarten Positionen um ein solches GT / AG liefern. Allerdings sind diese Positionen nicht so hoch konserviert wie die ersten und letzten beiden Intronpositionen.

9.3 Bisherige Arbeiten

Es stellt sich die Frage, welche Positionen links und rechts einer Spleisstelle noch einen Beitrag zur Identifizierung leisten und welche nicht mehr. Zur Beantwortung dieser Frage kann die bedingte Entropie eingesetzt werden. Dabei ermittelt man, ob es noch eine Abhängigkeit zwischen den Zeichen einer bestimmten Position und der Umgebung der Spleisstelle gibt. Ist dies nicht der Fall, dann braucht diese Position beim Erkennen von Spleisstellen nicht berücksichtigt zu werden.

Von Farach et al. [21] wurde folgender Versuch durchgeführt. Aus dem in Kapitel 8.1 erwähnten Datensatz, der 659 Introns und 669 Exons enthielt, wurden alle GT Dinukleotide sowie jeweils 30 Basenpaare links und rechts davon extrahiert. Diese 62 Zeichen langen Sequenzen wurden in zwei Klassen aufgeteilt, Donor GT’s und nicht Donor GT’s.

Zuerst soll die Menge der Donor GT Sequenzen betrachtet werden. Als R_n wird im Folgenden die Menge der Zeichen an Position n rechts von einem solchen

GT bezeichnet. R_1^{n-1} bezeichnet die Menge der Substrings, die die Zeichen 1 bis $n-1$ nach einem GT umfassen.

Jetzt kann man die bedingte Entropie $H(R_n|R_1^{n-1})$ ausrechnen. Dabei faßt man R_n und R_1^{n-1} als die Ausgabe zweier Quellen auf und schätzt die Wahrscheinlichkeiten der Zeichen bzw. Zeichenketten aus der jeweiligen Menge. $H(R_n|R_1^{n-1})$ gibt an, wie zufällig das n-te Zeichen nach einem GT vorkommt und wie groß seine Abhängigkeit von den vorangehenden $n-1$ Zeichen ist. Wenn $H(R_n|R_1^{n-1}) \approx 2$, dann ist R_n unabhängig von den vorherigen Zeichen und alle 4 Basen kommen an dieser Position etwa gleichhäufig vor. $H(R_n|R_1^{n-1})$ ist durch 2 beschränkt, da bei Unabhängigkeit $H(R_n|R_1^{n-1}) \leq H(R_n)$ gilt und $H(R_n)$ immer kleiner gleich 2 ist.

In [21] wird vorgeschlagen, daß $H(R_n|R_1^{n-1}) \approx 2$, wenn $H(R_n|R_1^{n-1}) > 1.8$. In diesem Fall hängt R_n nicht mehr von den Zeichen rechts der GT Stelle ab und liefert daher auch keine Information zur Identifikation der Spleisstelle. Andernfalls gibt es eine Abhängigkeit und man geht davon aus, daß diese Position einen Beitrag zur Erkennung der Donorstelle leistet.

Da die Menge der Donor GT's begrenzt ist und für große n keine vernünftigen Schätzungen der (bedingten) Wahrscheinlichkeiten mehr gemacht werden können, haben Farach et al. $H(R_n|R_1^{n-1})$ durch $H(R_n|R_1^3)$ approximiert.

$H(R_n|R_1^3)$ wurde dann wie folgt berechnet:

$$H(R_n|R_1^3) = \sum_{k \in K} \hat{p}(k) H(R_n|k),$$

wobei $K = {}'aaa', {}'aac', {}'aag', {}'aat', \ldots, {}'ttt'$ die Menge aller möglichen Substrings der Länge 3 ist und

$$H(R_n|k) = - \sum_{x \in \{A,C,G,T\}} \hat{p}(x|k)\, log_2(\hat{p}(x|k)).$$

Die bedingten Wahrscheinlichkeiten $\hat{p}(x|k)$ werden aus den Daten geschätzt und entsprechen der relativen Häufigkeit, mit der die drei Zeichen nach dem GT gleich k sind und das Zeichen x an Position n zu finden ist. Entsprechend gibt $\hat{p}(k)$ die relative Wahrscheinlichkeit an, mit der das Trinukleotid k rechts von einem Donor GT vorkommt. Für die Positionen $n = 1, 2, 3$ benötigt man keine Approximation und kann $H(R_n|R_1^{n-1})$ exakt berechnen. Für $n = 1$ berechnet man die normale Entropie $H(R_n)$, da R_1^0 leer ist.

Analog zu dem eben beschriebenen Schema kann man die bedingte Entropie der Zeichen links von einem Donor GT messen. Dazu haben R_n und R_1^{n-1} dieselbe Bedeutung wie oben, beziehen sich jetzt aber auf die Zeichen links eines GT's. Als Approximation wurde ebenfalls $H(R_n|R_1^3)$ verwendet.

Für Donorstellen berechneten Farach et al. $H(R_n|R_1^{n-1})$ für $n = 1, \ldots, 30$, sowohl für die linke, als auch die rechte Seite. Es zeigte sich, daß für die Positionen $1-3$ links und $1-4$ rechts vom GT die bedingten Entropien kleiner als 1.8 waren. Alle anderen Positionen zeigten keine Abhängigkeit von den 3 Zeichen links bzw. rechts des GT's und enthielten alle 4 Basen etwa gleichhäufig.

Zur Unterscheidung eines beliebigen und eines Donor GT's sollten also 3 Zeichen aus der linken und 4 aus der rechten Umgebung mit berücksichtigt werden. Dies ist konsistent mit den Ergebnissen von Stephens und Schneider 1992 [58], die allerdings nicht die bedingte Entropie, sondern die normale Entropie für alle Positionen ausrechneten. Dadurch wird gemessen, wie stark einzelne Positionen konserviert sind. Eine Abhängigkeit von der GT–Umgebung wird nicht berücksichtigt.

Anschließend wurde die bedingte Entropie der Positionen um nicht Donor GT's berechnet. Da die Menge an nicht Donor GT's wesentlich größer ist und genauere Wahrscheinlichkeitsschätzungen erlaubt, wurde $H(R_n|R_1^{n-1})$ durch $H(R_n|R_1^5)$ approximiert. Hier zeigt sich, daß alle Positionen links und rechts vom GT eine bedingte Entropie von mehr als 1.8 hatten. Das entspricht den Erwartungen, da nicht Donor GT's keine besondere Funktion besitzen, und daher sollten die umgebenden Zeichen auch keine Abhängigkeit oder Konservierung aufweisen.

Farach et al. haben außerdem versucht, dieses Experiment für Acceptorstellen durchzuführen. Hier gelang es allerdings nicht, Abhängigkeiten, wie diese bei Donorstellen zu finden sind, aufzudecken.

9.4 Untersuchung an größeren Datensätzen

Für die Untersuchung der Entropieunterschiede zwischen Exons und Introns hatte ich in Kapitel 8.2 einen großen Datensatz für verschiedene Organismen erstellt. Der Versuch von Farach et al. soll mit diesen Datensätzen wiederholt werden. Wie dieser Abschnitt zeigen wird, ermöglichen die größeren Datensätze das Finden von Abhängigkeiten in Donorstellen und auch in Acceptorstellen.

Für jede Spezies (Caenorhabditis, Drosophila, Bos taurus, Homo sapiens, Mus musculus, Arabidopsis thaliana, Oryza sativa, Zea mays) wurden aus dem Datensatz alle Donor GT's, nicht Donor GT's, Acceptor AG's und nicht Acceptor AG's mit den 30 Zeichen links und rechts extrahiert. Nach der oben beschriebenen Methode wurde $H(R_n|R_1^3)$ für $n = 1, \ldots, 30$ links und rechts von Donor– und Acceptorstellen berechnet. Für nicht Donor– und nicht Acceptorstellen wurde $H(R_n|R_1^5)$ ebenfalls für die 30 Zeichen links und rechts bestimmt. Die Verwen-

dung derselben Approximation stellt Vergleichbarkeit mit den Ergebnissen von Farach et al. sicher.

Die Ergebnisse für menschliche Donor und nicht Donor GT's sind in Abbildung 9.2 und 9.3 gezeigt und stimmen exakt mit den Ergebnissen von Farach et al. und Stephens und Schneider überein. Es zeigt sich, daß die Positionen 3 links und 4 rechts vom GT einen Wert von 1.78 haben, was sehr nahe an der Grenze von 1.8 liegt. Da die Datenmenge für menschliche Donor GT's sehr groß ist, können die Wahrscheinlichkeiten hinreichend exakt bestimmt werden. Diese beiden Positionen sollten also noch einen Beitrag zur Spleisstellenerkennung leisten, auch wenn dieser sehr klein ist. Abbildung 9.3 zeigt, daß es bei nicht Donor GT's solche Abhängigkeiten nicht gibt.

Für menschliche Acceptorstellen ergibt sich ein etwas anderes Bild (Abb. 9.4). Auf der Intronseite haben die Positionen 1 bis 14 einen Wert kleiner als 1.8. Dies korrespondiert zu den erwähnten Cytosin– und Thyminreichen Enden der Introns. Auf der Exonseite scheint nur die erste Position zur Acceptoridentifikation beizutragen. Für nicht Acceptor AG's (Abb. 9.5) haben wieder alle Positionen eine bedingte Entropie von nahe 2.

Die Kurven zeigen, daß die meiste Information der Spleisstellen im Intron zu finden ist. Dies hat wahrscheinlich den Sinn, die Codonabfolge des Exons möglichst wenig zu beeinflussen. Weiterhin haben Acceptorstellen mehr konservierte Positionen als Donorstellen. Von Stephens und Schneider [58] wurde gezeigt, daß Acceptorstellen eine eindeutige Identifizierung innerhalb eines transkribierten Gens erlauben, Donorstellen hingegen nicht. Dies könnte eventuell damit erklärt werden, daß zuerst die Acceptorstelle erkannt wird und dadurch die potentiellen Donorstellen so stark eingeschränkt werden, daß die Information der Donorstellen jetzt zur Identifikation ausreicht.

In Abbildung 9.6 wird die bedingte Entropie der Donorstellen für die anderen Organismen gezeigt. Für alle Spezies ergibt sich ein ähnliches Bild. So hat außer bei Arabidopsis die Position 3 rechts vom GT immer eine niedrigere Entropie als die Positionen 1 und 2. Bei den Pflanzen ist dies aber weniger ausgeprägt. Das bei Bos taurus, Caenorhabditis, Arabidopsis und besonders bei Zea mays die weiter entfernten Positionen teilweise eine bedingte Entropie unter 1.8 haben, ist meiner Meinung nach auf zu kleine Datensätze zurückzuführen und nicht auf Funktionen dieser Positionen bei der Spleisstellenerkennung. Die Maus–Donorstellen zeigen ein fast identisches Bild im Vergleich zum Mensch.

Die Acceptorstellen sind ebenfalls bei fast allen Spezies recht ähnlich (Abb. 9.7). Auffallend ist, daß bei allen Organismen außer Caenorhabditis die Position 2 links vom AG offenbar keine Rolle bei der Erkennung spielt. Auch

hier bewirken Datensätze mit geringer Größe das einige Kurven stark schwanken. Dies führt dazu, daß man bei Positionen, die eine bedingte Entropie von rund 1.8 haben, nicht mit Sicherheit eine Beteiligung an der Erkennung bestätigen oder ausschließen kann (z.B. Arabidopsis Position -8). Bei großen Datenmengen ergeben sich glattere Kurven (Maus und Mensch). Bei Zea mays impliziert die Kurve eine Beteiligung fast aller Positionen an der Identifizierung, was wahrscheinlich falsch ist. Ob bei Oryza sativa nur die Positionen -1 und -3 auf der Intronseite eine Rolle spielen, ist ebenfalls fraglich, aber nicht ausgeschlossen, da bekannt ist, daß bei Pflanzen der Spleissprozeß etwas verschieden zum Spleissvorgang in Wirbeltieren ist [8]. Auch wenn zu einer exakteren Bestimmung der relevanten Positionen größere Datenmengen benötigt werden, zeigen Abbildung 9.6 und 9.7, daß Spleisstellen bei vielen Organismen konserviert sind.

Die Kurven für die nicht Donor GT's und nicht Acceptor AG's der anderen Spezies zeigen alle keine abhängigen und konservierten Positionen und werden deshalb nicht extra gezeigt.

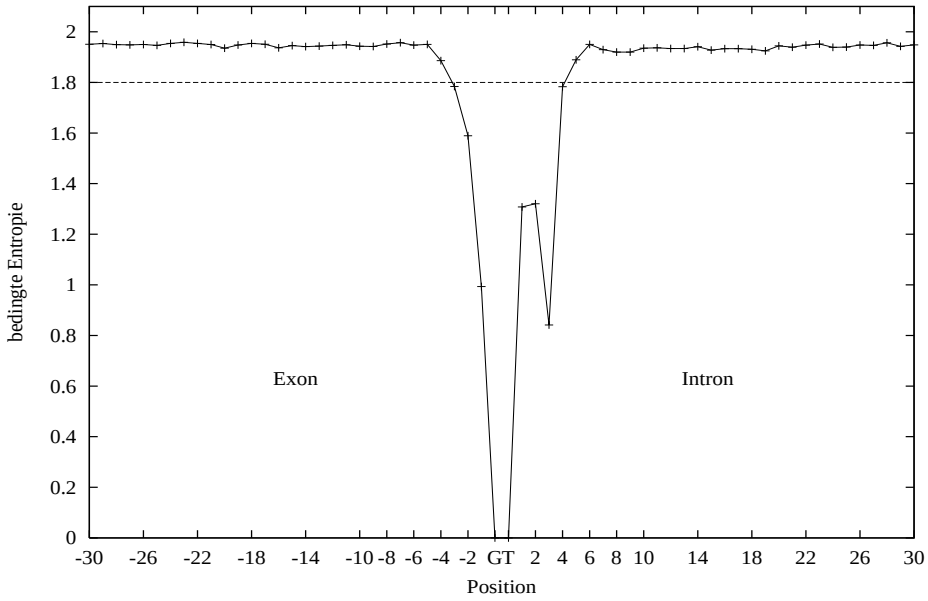


Abbildung 9.2: Homo Sapiens – Donor GT's

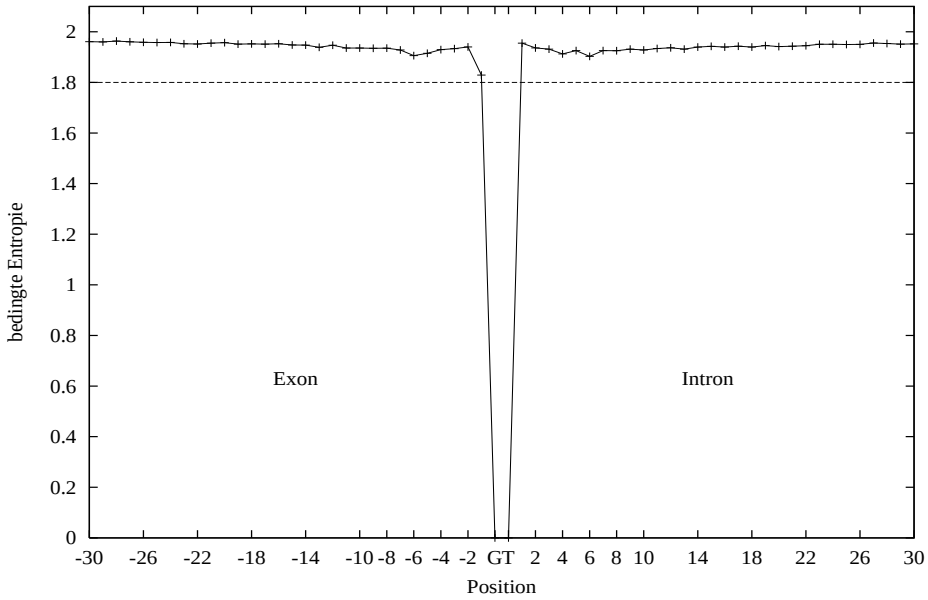


Abbildung 9.3: Homo Sapiens – Nicht Donor GT's

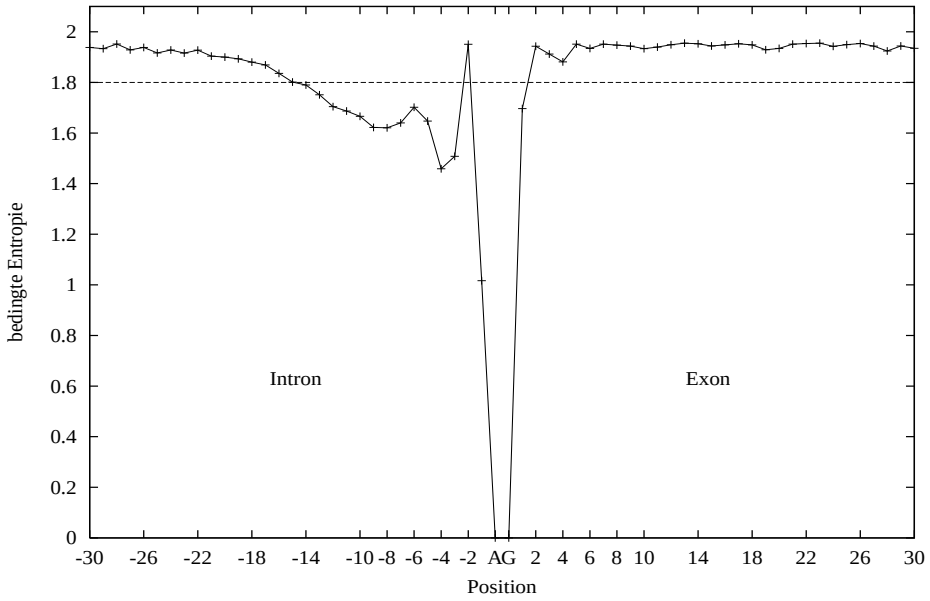


Abbildung 9.4: Homo Sapiens – Acceptor AG's

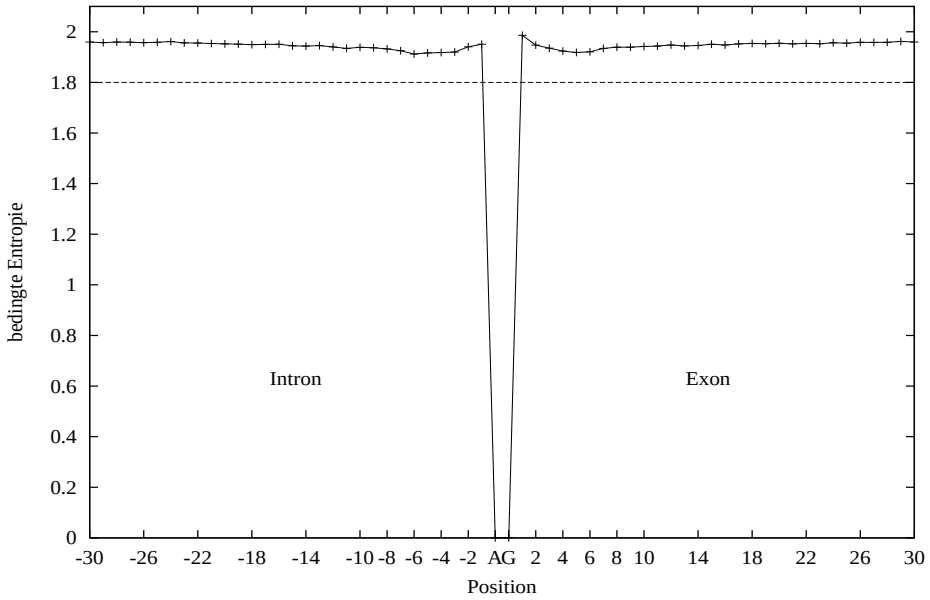


Abbildung 9.5: Homo Sapiens – Nicht Acceptor AG's

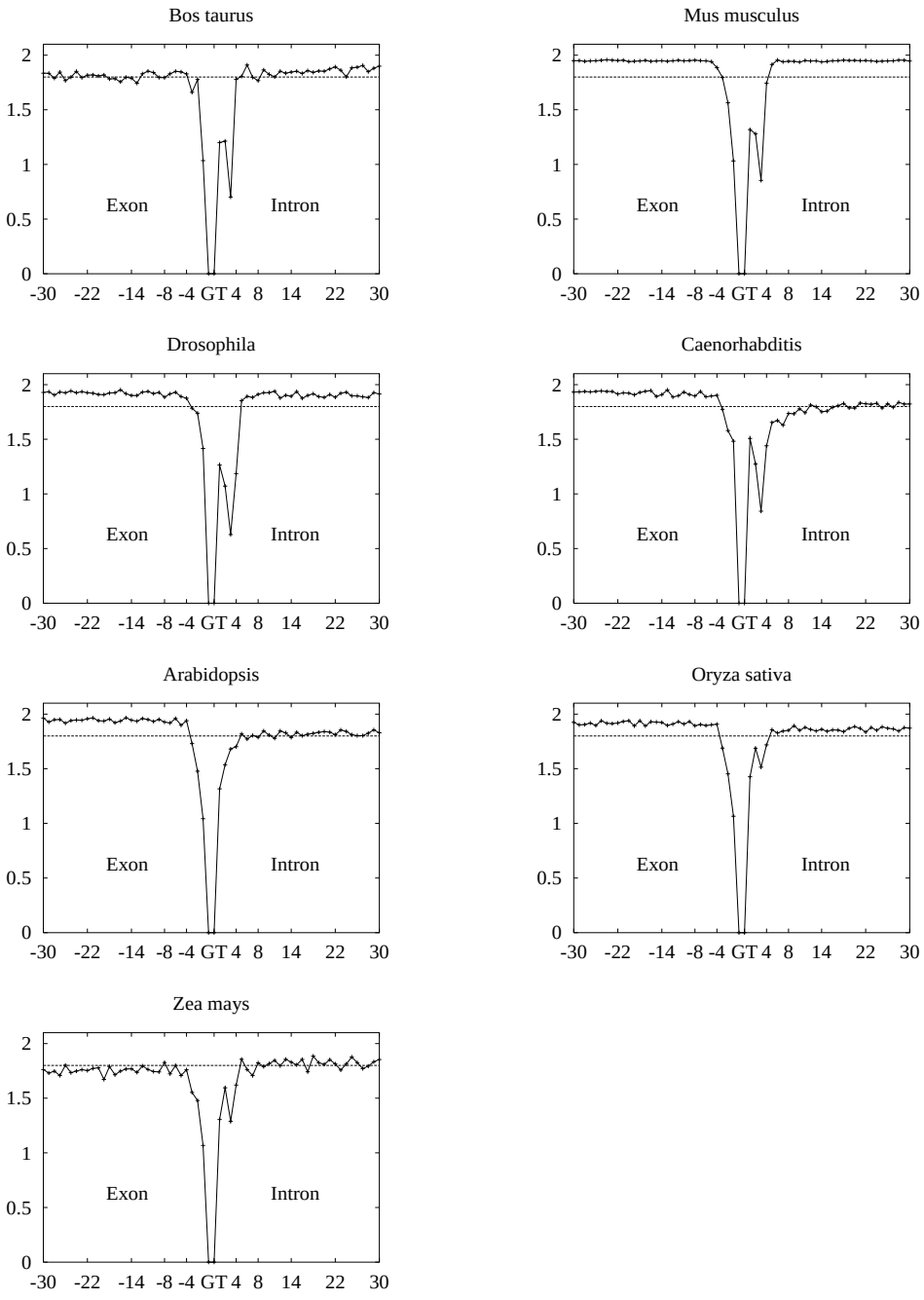


Abbildung 9.6: Donor GT Stellen von anderen Spezies

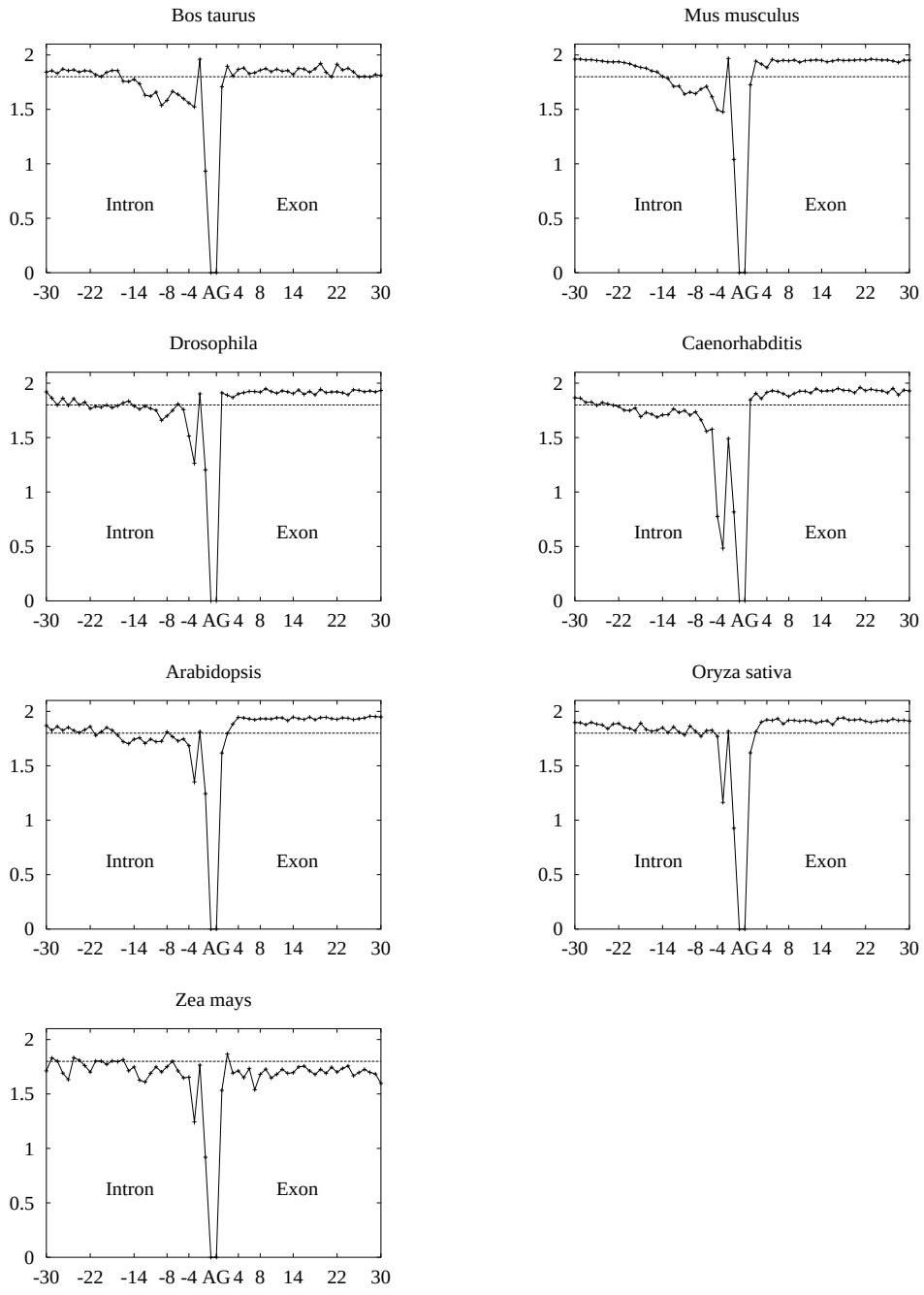


Abbildung 9.7: Acceptor AG Stellen von anderen Spezies

9.5 Zusammenfassung

In diesem Kapitel wurde die bedingte Entropie benutzt, um zu untersuchen, welche Positionen in der Umgebung von Spleisstellen eine Rolle bei der Erkennung spielen. Dafür wurde der Versuch von Farach et al. an einem größeren Datensatz für den Mensch sowie an Datensätzen für andere Spezies wiederholt. Für menschliche Donorstellen ergaben sich dieselben Ergebnisse. Ein Vergleich mit anderen Organismen zeigt Gemeinsamkeiten in den Positionen der Donorumgebung. Allerdings sind bei diesen Spezies, die Ergebnisse mit einer gewissen Unsicherheit behaftet, da sich aus den etwas kleineren Datensätzen die bedingten Wahrscheinlichkeiten nicht sehr genau schätzen lassen.

Im Gegensatz zu Farach et al. konnten mit Hilfe eines großen Datensatzes relevante Positionen in menschlichen Acceptorstellen identifiziert werden. Auch Acceptorstellen ergeben ähnliche Kurven für fast alle untersuchten Organismen, wobei es auch hier Schwankungen in den Kurven gibt, die wahrscheinlich auf eine ungenaue Wahrscheinlichkeitsschätzung zurückzuführen sind.

Kapitel 10

Zusammenfassung und Ausblicke

In dieser Diplomarbeit wurde untersucht, warum Standardkompressionsverfahren bei DNA Sequenzen keine oder nur eine sehr geringe Kompression erreichen. Der Grund dafür ist, daß Standardkompressoren Charakteristiken der menschlichen Sprache verwenden. Da es wesentliche Unterschiede zwischen DNA Sequenzen und menschlicher Sprache gibt, nutzen diese Verfahren fast keine charakteristischen Eigenschaften der DNA und erreichen deshalb auch keine Kompression.

Mit speziellen DNA Kompressoren können auch DNA Sequenzen komprimiert werden. Diese Verfahren berücksichtigen die Redundanz, die in inexakten Matches enthalten ist und kodieren Wiederholungen, die beliebig weit weg und lang sein können. Außerdem werden umgekehrte Komplemente erkannt. Die besten Methoden wurden in Kapitel 6 vorgestellt und ihre Leistung verglichen.

Trotzdem ist und bleibt die Kompression von DNA sehr schwierig. Nur in Ausnahmefällen erreichen die vorgestellten Verfahren Kompressionsraten von über 50 %. So liegen die Kompressionsraten der Benchmarksequenzen im Durchschnitt bei etwa 14 % und damit wesentlich niedriger als Kompressionsraten von englischem Text. Weiterhin benötigen die Kompressoren sehr viel Rechenzeit, da in der gesamten, bisher kodierten Sequenz nach (inexakten) Matches gesucht werden muß, während z.B. LZ77 sich für die Suche auf ein Fenster fester Länge beschränkt. Für lange Sequenzen benötigen die Programme außerdem sehr viel Arbeitsspeicher. Aus diesen Gründen finden DNA spezifische Kompressionsverfahren in der Praxis keine Anwendung beim Einsparen von Speicherplatz auf Medien.

Aber es gibt andere Bereiche in denen DNA Kompressoren erfolgreich eingesetzt werden können. Diese Bereiche betreffen biologische Probleme, für deren Lösung Methoden aus der Informationstheorie angewendet werden. So kann die

3D Struktur kurzer DNA Stücke mit einem Entropieschätzer bestimmt werden. Weiterhin kann ein Kompressor zwischen zufälligen und signifikanten Tandemwiederholungen unterscheiden.

Die bedingte Kompression ermöglicht die Erstellung eines Distanzmaßes für komplette Genome, das evolutionäre Ereignisse, wie Wiederholungen und umgekehrte Komplemente und auch nichtkodierende Regionen mit berücksichtigt. Von Li et al. [32] wurde gezeigt, daß dieses Distanzmaß zur Rekonstruktion von phylogenetischen Bäumen verwendet werden kann. Wie der in Kapitel 7 beschriebene Versuch aber zeigte, läßt sich ein genauer Abstand nur ermitteln, falls die Genome nicht zu sehr verschieden sind. Andernfalls ist keine Aussage über die Zeit seit der Aufspaltung einzelner Arten möglich, auch wenn sich die Grobstruktur des Stammbaumes noch einigermaßen rekonstruieren läßt. Es bleibt zu untersuchen, ob vielleicht ein DNA spezifischer Entropieschätzer, der keinen Overhead für exakte Rekonstruktion benötigt, auch bei stark unterschiedlichen Genomen eine exaktere Distanzschätzung liefern kann.

Unterschiedliche Regionen in einem Gen können mit Hilfe der Entropie charakterisiert werden, wie das für menschliche Exons und Introns von Farach et al. [21] gezeigt wurde. In Kapitel 8 wurde dieser Versuch für einen größeren Datensatz von menschlichen Exons und Introns wiederholt, wobei zusätzlich auch andere Spezies betrachtet wurden. Für alle Spezies ergab sich eine durchschnittlich höhere Entropie für Exonsequenzen. Weiterhin haben Exons in den meisten Fällen eine höhere Entropie als das Nachbarintron. Diese Ergebnisse wurden verwendet, um aus allen Vorhersagen eines Exonprediktors Teilmengen auszuwählen, die eine höhere Anzahl an korrekten Exons besaßen, als die Ausgangsmenge. Allerdings gelang es nicht, Entropieunterschiede zum Ausschluß von falsch positiven Vorhersagen zu verwenden, was für die Erhöhung der Spezifität ebenfalls hilfreich wäre. Interessant wäre es, ein Genvorhersage Programm zu entwickeln, das schon bei der Vorhersage der Exons Entropieunterschiede mit verwendet. Dazu könnte man die hier untersuchten Heuristiken in geeigneter Weise in das Entscheidungsschema vorhandener Programme integrierten. Eventuell erreicht man dadurch exaktere Vorhersagen.

Das letzte in dieser Diplomarbeit untersuchte Thema betrifft die Frage, welche Positionen in der Umgebung von Spleisstellen zu deren Erkennung im Organismus beitragen (Kap. 9). So haben hoch konservierte Positionen wahrscheinlich eine wichtige Aufgabe bei der Erkennung von Spleisstellen, da viele Mutationen in diesen Positionen ausselektiert wurden. Auch hier kann ein informationstheoretisches Maß, die bedingte Entropie, verwendet werden, um zu berechnen, wie hoch der Informationsgehalt oder die Konservierung einzelner Positionen ist. An-

hand der Datensätze aus Kapitel 8 konnten für menschliche Donor– und Acceptorstellen relevante Positionen identifiziert werden. Da diese Methode eine genaue Schätzung von bedingten Wahrscheinlichkeiten voraussetzt, sind große Datenmengen notwendig, die nur bei Mensch und Maus gegeben waren. Für die anderen Spezies lassen die Ergebnisse daher keine gesicherte Aussage zu, welche Positionen wirklich am Spleissvorgang beteiligt sind und welche nicht. Wenn mehr annotierte Gene zur Verfügung stehen, sollten sich mit dieser Methode ebenfalls alle relevanten Positionen finden lassen.

Literaturverzeichnis

[1] L. Allison, T. Edgoose, and T. I. Dix. Compression of strings with approximate repeats. In *Proceedings of International Conference on Intelligent Systems for Molecular Biology*, pages 8 – 16, Montreal, Canada, June 1998.

[2] L. Allison, D. Powell, and T. I. Dix. Compression and approximate matching. *Computer Journal*, 1(42):1 – 10, 1999.

[3] Datensatz annotierter Gene. ftp://www1.imim.es/pub/software/ace/testsets/testsets.tar.gz.

[4] A. Apostolico and S. Lonardi. Compression of biological sequences by greedy off-line textual substitution. In J. A. Storer and M. Cohn, editors, *Proceedings of IEEE Data Compression Conference*, pages 143 – 152, Snowbird, Utah, 2000. Also TR 99-037, Dept. of Computer Sciences, Purdue University.

[5] GenBank Bakteriengenome. http://www.ncbi.nlm.nih.gov/cog/.

[6] M. Borodovsky and J. McIninch. GENMARK: Parallel gene recognition for both DNA strands. *Comput. Chem.*, 17:123 – 133, 1993.

[7] B. Brejova, C. DiMarco, T. Vinar, S. Hidalgo, G. Holguin, and C. Patten. Finding patterns in biological sequences. Technical Report CS-2000-22, University of Waterloo, December 2000.

[8] J. Brown and C. Simpson. Splice site selection in plant pre-mRNA splicing. *Annual Rev. Plant Physiol. Plant Mol. Biol.*, 49:77 – 95, 1998.

[9] C. Burge and S. Karlin. Prediction of complete gene structure in human genomic DNA. *Journal of Molecular Biology*, 268:1 – 17, 1997.

[10] C. Burge and S. Karlin. Finding the genes in genomic DNA. *Current Opinion in Structural Biology*, 8:346 – 354, 1998.

[11] M. Burset and R. Guigo. Evaluation of gene structure prediction programs. *Genomics*, 34(3):353 – 367, 1996.

[12] X. Chen, S. Kwong, and M. Li. A compression algorithm for DNA sequences and its applications in genome comparison. In *Proceedings of the 10th Workshop on Genome Informatics (GIW'99)*, pages 52 – 61, December 1999.

[13] J.M. Claverie. Computational methods for the identification of genes in vertebrate genomic sequences. *Human Molecular Genetics*, 6(10):1735 – 1744, 1997.

[14] J. Cleary, R. Neal, and I. Witten. Arithmetic coding for data compression. *Communications of the ACM*, 30:520 – 540, 1987.

[15] J. Cleary and I. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 32(4):396 – 402, 1984.

[16] EMBL Nukleotid Datenbank.
http://www.ebi.ac.uk/embl/index.html.

[17] GenBank Nukleotid Datenbank.
http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=nucleotide.

[18] A. Dempster, N. Laird, and D. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society*, 39:1 – 38, 1977.

[19] Phylip Package Download.
http://evolution.genetics.washington.edu/phylip.html.

[20] QCCPack Download. http://qccpack.sourceforge.net/.

[21] M. Farach, M. Noordewier, S. Savari, L. Shepp, A. Wyner, and J. Ziv. On the entropy of DNA: Algorithms and measurements based on memory and rapid convergence. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 48 – 57, 1994.

[22] W. Fitch and E. Margoliash. The construction of phylogenetic trees. *Science*, 155:279 – 284, 1967.

[23] S. Grumbach and F.Tahi. Compression of DNA sequences. In *Proceedings of the IEEE Symposium on Data Compression*, pages 340 – 350, 1993.

[24] S. Grumbach and F.Tahi. A new challenge for compression algorithms: Genetic sequences. *Information Processing and Management*, 30:875 – 886, 1994.

[25] R. Guigo, S. Knudsen, N. Drake, and T. Smith. Prediction of gene structure. *Journal of Molecular Biology*, 266:141 – 157, 1992.

[26] D. Gusfield. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, 1997.

[27] D. Haussler. Computational genefinding. *Trends Biochem. Sci.*, 1998.

[28] GenCompress Implementierung.
http://cytosine.cs.ucsb.edu:8080/gencomp-linux.zip.

[29] MZEF Implementierung.
ftp://phage.cshl.org/pub/science/mzhanglab/mzef/human/.

[30] PPM Implementierung.
ftp://ftp.simtel.net/pub/simtelnet/win95/compress/ppmdg.zip.

[31] K. Lanctot, M. Li, and E. Yang. Estimating DNA sequence entropy. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 409 – 418, 2000.

[32] M. Li, J.H. Badger, X. Chen, S. Kwong, P. Kearney, and H. Zhang. An information-based sequence distance and its application to whole mitochondrial genome phylogeny. *Bioinformatics*, 17(2):149 – 154, 2001.

[33] P. Lio, A. Politi, M. Buiatti, and S. Ruffo. High statistics block entropy measures of DNA sequences. *Journal of Theoretical Biology*, 180:151 – 160, 1996.

[34] D. Loewenstern. *Sequence Classification Learning using Methods Derived from Entropy Estimation*. PhD thesis, Rutgers University, 1999.

[35] D. Loewenstern, H. Hirsh, P. Yianilos, and M. Noordewier. DNA sequence classification using compression-based induction. *DIMACS Technical Report 95-04*, April 1995.

[36] D. Loewenstern and P. Yianilos. Significantly lower entropy estimates for natural DNA sequences. In J.A. Storer and M. Cohn, editors, *Proceedings of IEEE Data Compression Conference*, pages 151 – 161, Snowbird, Utah, March 1997.

[37] T. Matsumoto, K. Sadakane, and H. Imai. Biological sequence compression algorithms. In T. Takagi, S. Miyano, A. Dunker, and A. Konagaya, editors, *Genome Informatics 2000*, pages 43 – 52, 2000. Universal Academy Press.

[38] A. Milosavljevic and J. Jurka. Discovering simple DNA sequences by the algorithmic significance method. *CABIOS*, 9(4):407 – 411, 1993.

[39] A. Milosavljevic and J. Jurka. Discovery by minimal length encoding: a case study in molecular evolution. *Machine Learning*, 12:69 – 87, 1993.

[40] A. Moffat, R. Neal, and I. Witten. Arithmetic coding revisited. In *Proceedings of the IEEE Data Compression Conference*, 1995.

[41] S. Needleman and C. Wunsch. A general method applicable to the search for similarities in the amino acid sequences of two proteins. *Journal of Molecular Biology*, 48:444 – 453, 1970.

[42] M. Nelson. *The data compression book*. M&T Publishing Inc., 1992.

[43] C. G. Nevill-Manning and I. H. Witten. Protein is incompressible. In *Proceedings of IEEE Data Compression Conference*, pages 257 – 266, 1999.

[44] Image Library of Biological Macromolecules des Instituts für Molekulare Biotechnologie Jena. http://www.imb-jena.de/image.html.

[45] D. Powell, D. Dowe, L. Allison, and T.I. Dix. Discovering simple DNA sequences by compression. In *Pacific Symposium on Biocomputing '98*, pages 595 – 606, 1998.

[46] E. Rivals, M. Dauchet, J. P. Delahaye, and O. Delgrange. Compression and genetic sequences analysis. *Biochimie*, 78(4):315 – 322, 1996.

[47] E. Rivals, M. Dauchet, J. P. Delahaye, and O. Delgrange. Fast discerning repeats in DNA sequences with a compression algorithm. In *Proceedings of 8th Workshop on Genome Informatics (GIW 97)*, pages 215 – 226, 1997. Universal Academy Press.

[48] E. Rivals, J. P. Delahaye, M. Dauchet, and O. Delgrange. A guaranteed compression scheme for repetitive DNA sequences. In J. A. Storer and M. Cohn, editors, *Data Compression Conference*, page 453, Snowbird, Utah, 1996.

[49] E. Rivals, O. Delgrange, J. P. Delahaye, M. Dauchet, M. O. Delorme, A. Henaut, and E. Ollivier. Detection of significant patterns by compression algorithms: The case of approximate tandem repeats in DNA sequences. *CABIOS*, 13(2):131 – 136, 1997.

[50] K. Sadakane, T. Okazaki, and H. Imai. Implementing the context tree weighting method for text compression. In *IEEE Data Compression Conference*, pages 123 – 132, 2000.

[51] N. Saitou and M. Nei. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406 – 425, 1987.

[52] S. Salzberg, A. Delcher, K. Fasman, and J. Henderson. A decision tree system for finding genes in DNA. Technical Report 1997-03, Department of Computer Science, John Hopkins University, 1997.

[53] A. Schmitt and H. Herzel. Estimating the entropy of DNA sequences. *Journal of Theoretical Biology*, 188:369 – 377, 1997.

[54] C.E. Shannon. A mathematical theory of communication. *Bell System Tech. Journal*, 27:379 – 423,623 – 656, 1948.

[55] M. Singer and P. Berg. *Gene und Genome*. Spektrum Akademischer Verlag, 1992.

[56] V. Solovyev, A. Salamov, and C. Lawrence. Predicting internal exons by oligonucleotide composition and discriminant analysis of spliceable open reading frames. *Nucleic Acids Research*, 22:5156 – 5163, 1994.

[57] V. Solovyev, A. Salamov, and C. Lawrence. Identification of human gene structure using linear discriminant functions and dynamic programming. In

Proceedings of Third International Conference on Intelligent Systems for Molecular Biology, pages 367 – 375, 1995.

[58] R. Stephens and T. Schneider. Features of spliceosome evolution and function inferred from an analysis of the information at human splice sites. *Journal of Molecular Biology*, 228:1124 – 1136, 1992.

[59] DNA Net Website. http://www.dna-dna.net/.

[60] T. Welch. A technique for high performance data compression. *IEEE Computer*, 17(6):8 – 19, 1984.

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

[62] Y. Xu, R. Mural, M. Shah, and E. Uberbacher. Recognizing exons in genomic sequence using GRAIL II. *Genet. Eng.*, 16:241 – 253, 1994.

[63] M. Q. Zhang. Identification of protein coding regions in the human genome based on quadratic discriminant analysis. *Proceedings of the National Academy of Sciences*, 94:559 – 564, 1998.

[64] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337 – 343, 1977.

[65] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530 – 536, 1978.

[66] J. Ziv, A.D. Wyner, and A.J. Wyner. On the role of pattern matching in information theory. *IEEE Transactions on Information Theory*, 44:2045 – 2056, 1998.

Selbständigkeitserklärung

Hiermit erkläre ich, daß ich die vorliegende Arbeit eigenständig und unter ausschließlicher Benutzung der angegebenen Hilfsmittel angefertigt habe.

Leipzig, 5. Juni 2002

Michael Hiller