Analyse von Translationsstarts in prokaryotischen Genomen mit Methoden des Maschinellen Lernens

Dissertation
zur Erlangung des Doktorgrades
der Mathematisch-Naturwissenschaftlichen Fakultäten
der Georg-August-Universität zu Göttingen

vorgelegt von
Maike Tech
aus Lübeck

Göttingen, 2007

Referent : Prof. Dr. B. Morgenstern
Korreferent : Prof. Dr. S. Waack
Tag der Disputation : 2.11.2007

Danksagung

Als erstes danke ich Prof. Dr. Burkhard Morgenstern dafür, dass er mir ermöglicht hat, an seinem Lehrstuhl zu promovieren. Er war stets ein guter Chef und Ansprechpartner. Dr. Peter Meinicke danke ich für die gute Zusammenarbeit, die intensive Betreuung meiner Arbeit und sein Bemühen, mir die Geheimnisse der angewandten Statistik nahezubringen. Vor allem möchte ich mich dafür bedanken, dass ich mich bei Euch in allem unterstützt gefühlt habe. Ich danke außerdem meinen Kolleginnen und Kollegen von der biologischen und der medizinischen Bioinformatik für die gute Arbeitsatmosphäre und das ganze Drumherum. Thomas Lingner danke ich auch für die Hilfe beim Korrekturlesen. Für die Sicherstellung der Kaffee- und Keksversorgung, fürs Korrekturlesen und dafür, dass er sich immer geduldig meine »Sorgen« anhört, danke ich dem man-for-all-seasons, Jürgen Dönitz. Prof. Dr. Stephan Waack danke ich dafür, dass er mich mit seiner Vorlesung vor »langer Zeit« für die Informatik begeistert hat und für seine Anteilnahme an meinem Werdegang. Ebenso danke ich Dr. Rainer Merkl, der mich auf meinem Weg zur Bioinformatik begleitet hat. An dieser Stelle möchte ich noch einige weitere Personen nennen, die in dieser Hinsicht einen wesentlichen Einfluss auf meinen Leben hatten: Andreas P. Priesnitz, Jörg Helms, Nils Höltge und Matthias Homeister. Vielleicht wär alles anders gekommen, wenn ihr nicht gewesen wärt… An Andreas geht noch ein Extra-Dank für 1001 Diffs und den seelischen Beistand aus der Ferne. Ich danke auch meiner Familie, meinen Eltern, meiner Oma, Janina, Anton, Manu und Hugo, dem besten Freund aller Zeiten, dafür, dass sie immer für mich da sind und mir Sicherheit und Selbstvertrauen geben. Und schließlich bleiben noch zwei zu nennen, die aus meinem Leben nicht mehr wegzudenken sind und die mich immer daran erinnern, dass es noch sehr viel mehr gibt als arbeiten: Helena und Rasmus.

Inhaltsverzeichnis

Abkürzungen und fremdsprachliche Ausdrücke

3′-Ende	»Ende« einer Nukleotidsequenz (freistehende OH-Gruppe an Position 3 des Zuckermoleküls)
5′-Ende	»Anfang« einer Nukleotidsequenz (freistehende OH-Gruppe an Position 5 des Zuckermoleküls)
A	Adenin
BP	**B**ase **P**air (Basenpaar) ≡ nt (*nucleotide*)
C	Cytosin
C-Terminus	»Ende« der Aminosäuresequenz (mit freistehender Carboxyl-Gruppe)
COOH	Carboxyl-Gruppe
CDS	**Co**d**d**ing **S**equence (dt. kodierende Sequenz)
Chr.	Chromosom
default	Standard
DNA	**D**esoxyribo**n**ucleic **A**cid (dt: DNS – Desoxyribonukleinsäure)
downstream	bezogen auf die Leserichtung im DNA-Strang: stromabwärts, in 3′-Richtung
GenBank, GBK	Datenbank am NCBI
Input Space	Eingaberaum
Interface	Schnittstelle
IUPAC	**I**nternational **U**nion of **P**ure and **A**pplied **C**hemistry
G	Guanin
Feature Space	Merkmalsraum
FN	**F**alse **N**egatives (dt: Falsch-Negative)
FP	**F**alse **P**ositives (dt: Falsch-Positive)
FPR	**F**alse **P**ositives **R**ate (Rate der Falsch-Positiven)
Frame-Shift	Verschiebung im Leserahmen
in-frame	im gleichen Leserahmen
Label	Kennzeichnung einer Klasse (dt. Markierung)
LSQ	**L**east **S**quares **M**ethod
Motiv	kurzes Sequenzmuster
MBP	**M**ega**b**asen**p**aar ≡ Mio. nt (*nucleotide*)
NCBI	**N**ational **C**enter for **B**iotechnology **I**nformation
N-Gruppe	Amino-Gruppe (NH_2)

nt	Nucleotide (dt: Nukleotid) ≡ BP
N-Terminus	»Anfang« der Aminosäuresequenz (mit freistehender Amino-Gruppe)
OH	Hydroxy-Gruppe
ORF	**O**pen **R**eading **F**rame (offener Leserahmen)
overfitting	Überanpassung
postprocessing	Nachbearbeitung
PPV	**P**ositive **P**redictive **V**alue (dt. Positiver Vorhersagewert)
PRC	**P**recision **R**ecall **C**urve
RBS	**R**ibosome **B**inding **S**ite (dt: Ribosombindestelle)
RLSQ	**R**egularized **L**east **S**quares (classification)
RNA	**R**ibo**n**ucleic **A**cid (dt: RNS – Ribonukleinsäure)
Score	Quantitativer Gütewert
Screenshot	»Bildschirmfoto«
Shift	Verschiebung
TN	**T**rue **N**egatives (dt: Wahr-Negative)
Tool	Werkzeug, sinngemäß Software-Werkzeug
TP	**T**rue **P**ositives (dt: Wahr-Positive)
TPR	**T**rue **P**ositives **R**ate (Rate der Wahr-Positiven ≡ Sensitivität/100)
T	Thymin
upstream	bezogen auf die Leserichtung im DNA-Strang: stromaufwärts, in 5′-Richtung
U	Urazil

Notation

$\mathscr{A}$	Alphabet $\mathscr{A}$
$\mathscr{A}^K$	Wort der Länge K aus dem Alphabet $\mathscr{A}$
$\mathbb{R}$	Reelle Zahlen
$\mathcal{X}$	Eingaberaum (*input space*)
$\mathcal{F}$	Merkmalsraum (*feature space*)
$\phi : \mathcal{X} \mapsto \mathcal{F}$	Abbildung des Eingaberaumes $\mathcal{X}$ auf den Merkmalsraum $\mathcal{F}$
$\mathbf{x}$	bezeichnet einen Vektor $\mathbf{x} = [x_1,\ x_2,\ \ldots,\ x_d]^\top$
$\mathbf{x}_i$	bezeichnet den $i$ten Vektor $\mathbf{x}$
$\|\mathbf{x}\|$	1-Norm des Vektors $\mathbf{x} \in \mathbb{R}^d$: $\|\mathbf{x}\| = \sum_{i=1}^d \|x_i\|$
$\|\mathbf{x}\|^2$	quadratische Euklidische Norm des Vektors $\mathbf{x} \in \mathbb{R}^d$: $\|\mathbf{x}\|^2 = \sum_{i=1}^d \|x_i\|^2$
$\mathbf{x}^\top$	Transponierter Vektor $\mathbf{x}$, entsprechend für Matrizen
$\langle \mathbf{x}, \mathbf{x}' \rangle$	Inneres Produkt (Skalarprodukt) von $\mathbf{x}$ und $\mathbf{x}'$
$k(\mathbf{x}_i, \mathbf{x}_j)$	Kern
$\mathbf{K}$	Kernmatrix mit $k(\mathbf{x}_i,\ \mathbf{x}_j) = \mathbf{K}_{ij}$
E	Fehler-/Verlustfunktion
$\mathcal{T}$	Trainingsmenge
$\mathbf{S}$	Glättungsmatrix
$\mathbf{P}$	Wahrscheinlichkeitsmatrix
$\mathbf{W}$	Positionsgewichtsmatrix (PWM – *positional weights matrix*)
$\mathbf{X}$	Datenmatrix
$\forall$	für alle
e	Basis des Natürlichen Logarithmus
α	Gewichtsparameter
λ	Regularisierungsparameter
σ	Glättungsfaktor
∇f	Gradient von f

Kapitel 1

Einleitung

Die vorliegende Arbeit befasst sich mit der Analyse der Signale prokaryotischer Translationsstarts mit dem Ziel, die automatische Annotation von Genomen zu verbessern. Dabei kommen zwei im Rahmen dieser Arbeit entwickelte Ansätze aus dem Bereich des Maschinellen Lernens zum Einsatz: Der Oligo-Kern-Algorithmus [1], ein überwachtes Verfahren zur Analyse von Signalen in biologischen Sequenzen, und TICO (*Translation Initiation site COrrection*), ein Programm zur (Re-)Annotation von Translationsstarts mit einem unüberwachten Lernverfahren [2, 3, 4].

Es wird gezeigt, dass der Oligo-Kern-Algorithmus für die Analyse und Identifikation biologischer Signale gut geeignet ist. In einer Fallstudie zu Translationsstarts des Eubakteriums *Escherichia coli* K-12 wird belegt, dass der Oligo-Klassifikator eine hohe Performanz bei der Vorhersage auf experimentell verifizierten Daten aufweist. Eine Visualisierung der diskriminativen Merkmale ermöglicht eine biologisch sinnvolle Interpretation. Der Algorithmus ist flexibel, hinsichtlich der Länge der betrachteten Oligomere und des Grades an Positionsinformation, so dass er zur Analyse anderer biologischer Sequenzen angepasst werden kann.

Im zweiten Teil der Arbeit wurde ein Verfahren zur automatischen Reannotation von Genstarts entwickelt, das in dem Programmpaket TICO realisiert ist. Das Programm TICO erzielt eine signifikante Verbesserung der Vorhersage prokaryotischer Translationsstarts im Vergleich zu früheren Ansätzen. Dabei wird eine initiale Annotation, wie sie beispielsweise mit einem klassischen Genvorhersageprogramm erstellt werden kann, nachbearbeitet. Der Algorithmus bietet eine Visualisierungsfunktion, welche eine intuitive Darstellung der diskriminativen Merkmale ermöglicht. Das Programm ist über ein Web-Interface (Webschnittstelle) und als Kommandozeilenprogramm für Linux und Windows implementiert und frei verfügbar.

1.1 Biologischer Hintergrund

Das Auffinden und Charakterisieren von Genen in genomischen Sequenzen ist Voraussetzung für die Erforschung und das Verständnis der Physiologie und der Evolution des Lebens. Der gesamte Stoffwechsel, sowohl einzelner Zellen als auch höherer Organismen – als hochkomplexer Zusammenschluss vieler milliarden Zellen – ist im Genom kodiert. Die Entschlüsselung von Genomen und des Zusammenspiels von Genen ist somit Grundlage vieler Forschungsgebiete in Biologie, Biotechnologie und Medizin.

1.1.1 Aufbau der DNA

Das Genom eines Organismus besteht aus einer oder mehreren selbst-replizierenden Einheiten der DNA (*desoxyribonucleic acid*). Die DNA setzt sich aus vier Nukleotiden zusammen, welche jeweils aus einem Zucker (Desoxyribose), einer Base und einem Phosphatrest bestehen. Die Nukleotide unterscheiden sich durch ihre Base, diese wird als Synonym für das ganze Nukleotid verwendet: Adenin (A), Cytosin (C), Guanin (G) und Thymin (T). Die DNA liegt als Makromolekül in Form einer Doppelhelix, die aus zwei reverskomplementären Nukleotidketten besteht, in der Zelle vor. Die einzelnen Nukleotide sind über Phosphodiesterbindungen verkettet. Es wird jeweils die 5'-OH-Gruppe eines Zuckers über einen Phosphatrest mit der 3'-OH-Gruppe des nächsten Zuckers verknüpft. Die Doppelstrangbildung erfolgt durch Wasserstoffbrücken, die zwischen den Basen der Nukleotide ausgebildet werden. Komplementär sind dabei die Basen A und T, die über zwei Wasserstoffbrücken verbunden sind, sowie G und C, zwischen denen drei Wasserstoffbrücken ausgebildet werden (siehe Abbildung 1.1).

Ein vollständiger DNA-Doppelstrang eines Prokaryoten – als solche werden Bakterien und Archaeen zusammengefasst – wird als (Bakterien-)Chromosom bezeichnet. Während die Genome höherer Organismen (Eukaryoten) meist aus mehreren komplex aufgebauten Chromosomen besteht, die im Kern der Zelle liegen, umfasst das gesamte Genom eines Prokaryoten meist nur ein oder zwei circuläre Chromosomen und liegt direkt im Zytoplasma der Zelle. Im Fall prokaryotischer Organismen gibt es weitere selbst-replizierende DNA-Einheiten, die als Plasmide bezeichnet werden. Plasmide unterscheiden sich im Aufbau nicht von bakterieellen Chromosomen. Die Sequenz eines Plasmids ist aber in der Regel deutlich kürzer als die eines Chromosoms. Weiterhin ist ein Plasmid dadurch ausgezeichnet, dass auf ihm keine essentiellen Funktionen des Stoffwechsels kodiert sind. Durch die geringere Größe und den einfacheren Aufbau ist die genetische Manipulation prokaryotischer Genome im Allgemeinen weitaus einfacher als die Manipulation

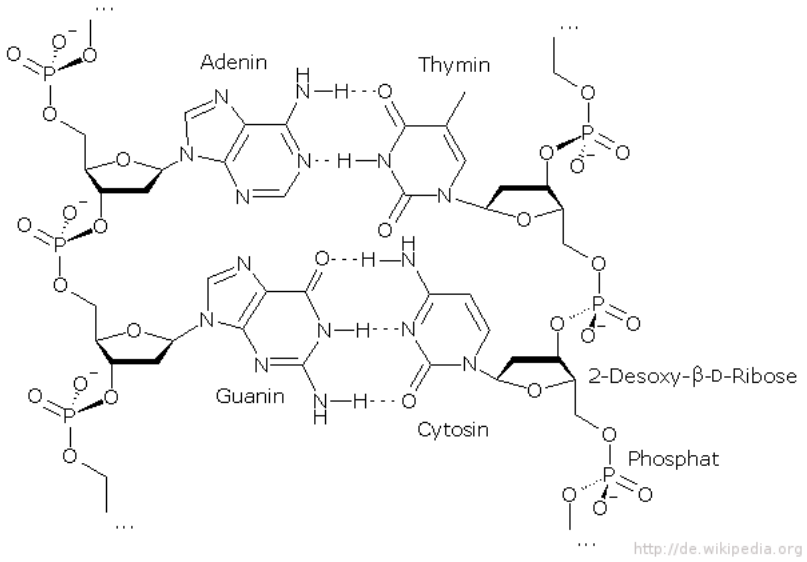

Abbildung 1.1: **Darstellung der DNA-Struktur. Die Wasserstoffbrücken (Mitte) verbinden jeweils zwei Basen. Die Basen (Adenin, Cytosin, Guanin, Thymin) sind an das »Rückgrat« aus Zuckenmolekülen gebunden.**

eukaryotischer Genome, weswegen ihnen eine große Bedeutung in der Biotechnologie zukommt.

Pro- und Eukaryoten unterscheiden sich in der Organisation der genetischen Information in einigen wesentlichen Punkten: Die kodierenden Bereiche eines Genes liegen bei Prokaryoten an einem Stück vor (*single exon*). Im Gegensatz dazu werden eukaryotische Gene aus mehreren kodierenden Exons zusammengesetzt, welche in der DNA durch nicht-kodierende Introns getrennt sind. Das Verhältnis kodierender Sequenzbereiche zu nicht-kodierenden ist ein weiterer wichtiger Unterschied. Es wird angenommen, dass in prokaryotischen Genomen mehr als 80% der Sequenz kodierend sind, während in eukaryotischen von etwa 10% ausgegangen wird. Daher ist das Problem, Gene in prokaryotischen Genomen zu finden, deutlich weniger schwierig als in eukaryotischen Genomen. Schon die Gesamtmenge der zu durchsuchenden Daten ist für Prokaryoten wesentlich geringer. Zum Vergleich: Prokaryotische Genome umfassen einige millionen Nukleotide[1] mit durchschnittlich 4000 Genen, während eukaryotische viele milliarden Nukleotide mit mehreren 10 000 Genen umfassen können. In vielen Bereichen erzielt die automatisierte Genvorhersage für prokaryotische Genome daher schon sehr gute Ergebnisse. Dennoch gibt es Bereiche, in denen die Genvorhersage für prokaryotische Sequenzen bislang nicht zufriedenstellend gelöst ist, worauf in Abschnitt 1.2 (S. 8) näher eingegangen wird.

[1]Die Angabe der Länge von Genomsequenzen erfolgt meist in der Einheit Basenpaar (BP), wobei ein Basenpaar einem Nukleotid entspricht.

Die meisten Gene in prokaryotischen Genomen kodieren für Proteine. Daher wird das Wort *kodieren* im Folgenden bis auf Weiteres in diesem Sinne verwendet, ebenso werden als Gene nur solche bezeichnet, die für ein Protein kodieren. Wenn Gene gemeint sind, die nicht für ein Protein kodieren, so werden sie explizit als RNA-Gene (*ribonucleic acid*) bezeichnet.

Da die vorliegende Arbeit sich ausschließlich mit der Genvorhersage in prokaryotischen Organismen befasst, beziehen sich alle folgenden Ausführungen nur auf die prokaryotische Genetik.

1.1.2 Von der DNA zum Protein

Ein Protein ist ein Makromolekül, dass aus einer Sequenz von Aminosäuren besteht. Die Aminosäuresequenz eines Proteins ist im Genom des Organismus kodiert. Die Kodierung basiert auf einer Folge von nicht-überlappenden Nukleotid-Tripletts, diese werden als Codons bezeichnet. Aus Sicht der Bioinformatik ist ein Codon definiert als ein Wort der Länge drei aus dem Alphabet $\mathscr{A} = \{A,\ T,\ G,\ C\}$.

Der Vorgang der Übersetzung einer DNA-Sequenz in die Aminosäuresequenz eines Proteins (Expression, siehe Abbildung 1.2, S. 5) erfolgt in zwei Schritten. Zunächst wird bei der Transkription durch die RNA-Polymerase eine »Kopie« des kodierenden DNA-Strangs erstellt, die mRNA (*messenger ribonucleic acid*). Diese unterscheidet sich von der DNA durch das Zuckermolekül (Ribose statt Desoxyribose) und durch die Base Urazil (U), welche an Stelle von Thymin verwendet wird. Die mRNA liegt als lineares Makromolekül im Zytoplasma der Zelle, so dass jeweils auf der einen Seite eine 5'-OH-Gruppe frei steht (5'-Ende) und auf der anderen eine 3'-OH-Gruppe (3'-Ende). Sie wird bei der Translation durch Ribosomen in die Aminosäuresequenz übersetzt. Die Ableserichtung ist dabei immer vom 5'-Ende zum 3'-Ende. Das Ribosom bindet dazu auf einem etwa 5-9 BP langen Abschnitt der mRNA etwa 10 BP vor dem Startcodon (*upstream*, in 5'-Richtung), der Ribosombindestelle (RBS – *ribosome binding site*). Diese ist komplementär zum 3'-Ende der 16S-rRNA, einer Untereinheit des Ribosoms, und ist in vielen systematischen Gruppen von Prokaryoten hochkonserviert.

Bei der Translation wird jedem Codon eine bestimmte Aminosäure zugeordnet. Ausnahme bilden hierbei nur die Stoppcodons (in der Regel TAA, TAG und TGA). Sie kodieren nur für einen Translationsstopp und kommen daher nicht innerhalb, sondern nur am Ende eines Genes vor. Nur in seltenen Ausnahmefällen kodiert ein Stoppcodon für eine Aminosäure, beispielsweise kann das Codon TGA in mitochondrialer DNA für die Aminosäure Tryptophan kodieren. In solchen Fällen kodiert das betreffende Codon dann nicht für einen Translationsstopp.

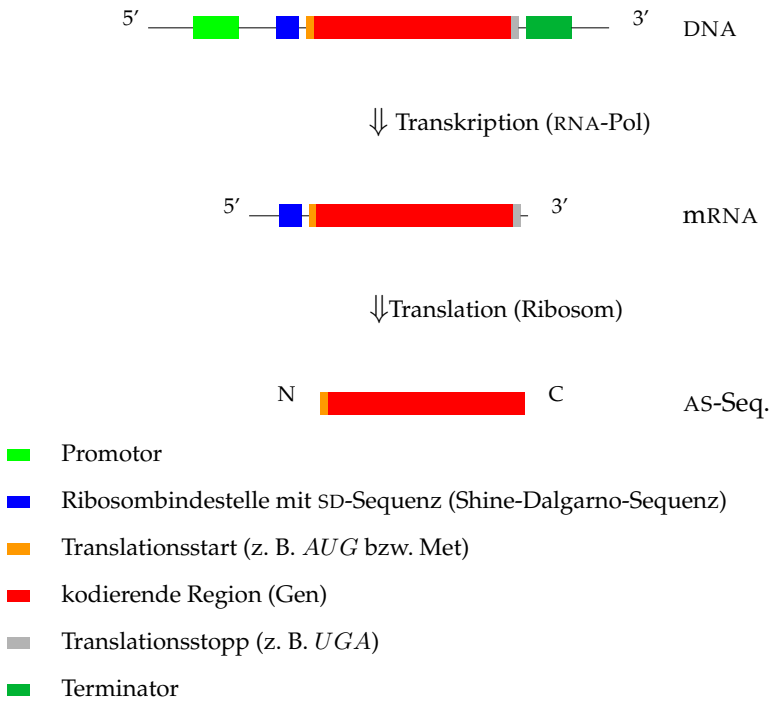

Abbildung 1.2: **Schematische Darstellung der Genexpression bei Prokaryoten. Die Transkription beginnt mit der Bindung der RNA-Polymerase (RNA-Pol) am Promotor. Sie erstellt eine »Kopie« des kodierenden Strangs, die *messenger*RNA (mRNA). Es folgt die Translation: Das Ribosom bindet an die mRNA und übersetzt diese in eine Aminosäuresequenz (AS-Seq.). Transkription und Translation beginnen immer am 5′-Ende der Sequenz. Die Orientierung des Proteins wird mit N- bzw. C-Terminus angegeben.**

Eine besondere Funktion bei der Translation haben auch die Startcodons, in den meisten Fällen ist dies ATG (über 90%), weiterhin kommen noch GTG, TTG und selten CTG oder andere vor. Sie markieren den Start eines Gens und kodieren gleichzeitig für eine Aminosäure, z. B. kodiert ATG für die Aminosäure Methionin. Da die Startcodons für eine Aminosäure kodieren, können sie sowohl am Translationsstart als auch innerhalb von Genen vorkommen. Der »Anfang« der Proteinsequenz wird als N-Terminus bezeichnet, da hier eine Amino-Gruppe freisteht. Das »Ende« wird entsprechend als C-Terminus bezeichnet. Da das Startcodon ATG am häufigsten auftritt, ist also am N-Terminus eines Proteins meist ein Methionin zu finden.

Prokaryotische Genome sind in Form von *Operons* organisiert. Ein Operon besteht aus einem oder mehreren Genen, die sukzessiv meist mit kurzem Abstand in der gleichen Leserichtung liegen. Diese Gene stehen alle unter Kontrolle eines Promotors, d. h. sie werden gemeinsam reguliert und sie werden gemeinsam transkribiert, es resultiert

daraus also eine mRNA für alle Gene eines Operons. Die Trennung der Produkte erfolgt erst bei der Translation. Das hat zur Folge, dass sich die Signale der Translationsinitiation von Genen am »Anfang« eines Operons (*operon leader*) von denen »innerhalb« eines Operons unterscheiden können.

1.1.3 Der genetische Kode

Aus den vier Buchstaben des genetischen Alphabets können 64 Wörter der Länge drei gebildet werden. Es werden insgesamt 20 Aminosäuren kodiert, d. h. es gibt mehr als dreimal so viele Codons wie Aminosäuren. Die meisten Aminosäuren werden durch mehr als ein Codon kodiert (*synonyme* Codons), was als »Degeneriertheit des genetischen Kodes« bezeichnet wird (siehe Tabelle 1.1, S. 7). Dadurch können Gene, die sich auf Nukleotidebene unähnlich sind, für eine ähnliche Aminosäuresequenz kodieren.

Verschiedene Organismen und systematische Gruppen zeigen unterschiedliche Präferenzen bei der Verwendung von Codons (*Codon-Usage*). Ebenso kann sich die Nukleotidzusammensetzung der DNA stark unterscheiden. Der Gehalt an Guanin und Cytosin (GC-Gehalt) wird daher auch als Kenngröße für Genome angegeben. Er variiert in prokaryotischen Genomen im Bereich von 25% bis 75%.

Auch innerhalb eines Genoms schwanken GC-Gehalt und Codon-Usage zum Teil stark. In Regionen vor Genstarts liegt der GC-Gehalt häufig deutlich unterhalb des Mittelwerts des restlichen Genoms, was damit erklärt werden kann, dass hier die beiden DNA-Stränge für die Transkription »aufgeschmolzen« werden müssen. Da zwischen A und T nur zwei Wasserstoffbrücken gebildet werden, ist die Energie, die zur Trennung der beiden Stränge nötig ist, in AT-reichen Regionen geringer. Schwankungen in GC-Gehalt und Codon-Usage können auch in der unterschiedlichen Expressionsrate der Gene begründet sein [5]. Dies wird mit einer stärkeren Codon-Selektion in hoch-exprimierten Genen erklärt. Ein weiterer Grund für signifikantes Abweichen in GC-Gehalt und Codon-Usage innerhalb eines Genoms liegt im lateralen Gentransfer, ein Phänomen, das im Reich der Prokaryoten sehr verbreitet ist. Gruppen von Genen können so aus dem Genom eines Organismus in das eines anderen einwandern. Die eingewanderten Gene werden dann als Fremdgene (*alien genes*) bezeichnet. Der laterale Gentransfer kann zur Folge haben, dass ganze Bereiche (Genomische Inseln – *genomic islands*) stark vom genomischen Durchschnitt abweichen [6, 7]. Solche Bereiche bereiten der automatisierten Genvorhersage immer noch Schwierigkeiten.

Durch die Kodierung in Form von Codons, können die Gene, je nachdem an welcher Position sie beginnen, sechs verschiedenen »Leserahmen« zugeordnet werden – drei im

Tabelle 1.1: **Degeneriertheit des genetischen Kodes. In der Tabelle sind die Codons und in Klammern die jeweils von ihnen kodierten Aminosäuren angegeben (Nomenklatur nach IUPAC*). Die Startcodons sind in rot angegeben, die Stoppcodons in blau.**

		2					
		T/U	C	A	G		
1	T/U	TTT (Phe)	TCT (Ser)	TAT (Tyr)	TGT (Cys)	T/U	**3**
		TTC (Phe)	TCC (Ser)	TAC (Tyr)	TGC (Cys)	C	
		TTA (Leu)	TCA (Ser)	TAA (Stopp)	TGA (Stopp)	A	
		TTG (Leu)	TCG (Ser)	TAG (Stopp)	TGG (Trp)	G	
	C	CTT (Leu)	CCT (Pro)	CAT (His)	CGT (Arg)	T/U	
		CTC (Leu)	CCC (Pro)	CAC (His)	CGC (Arg)	C	
		CTA (Leu)	CCA (Pro)	CAA (Gln)	CGA (Arg)	A	
		CTG (Leu)	CCG (Pro)	CAG (Gln)	CGG (Arg)	G	
	A	ATT (Ile)	ACT (Thr)	AAT (Asn)	AGT (Ser)	T/U	
		ATC (Ile)	ACA (Thr)	AAC (Asn)	AGC (Ser)	C	
		ATA (Ile)	ACA (Thr)	AAA (Lys)	AGA (Arg)	A	
		ATG (Met)	ACG (Thr)	AAG (Lys)	AGG (Arg)	G	
	G	GTT (Val)	GCT (Ala)	GAT (Asp)	GGT (Gly)	T/U	
		GTC (Val)	GCC (Ala)	GAC (Asp)	GGC (Gly)	C	
		GTA (Val)	GCA (Ala)	GAA (Glu)	GGA (Gly)	A	
		GTG (Val)	GCG (Ala)	GAG (Glu)	GGG (Gly)	G	

*International Union of Pure and Applied Chemistry

(+)-Strang, drei im reverskomplementären (-)-Strang. Für einen Strang wird der Leserahmen eines Gens oft als Modulo 3 der Position des Starts bzw. des Stopps definiert.

Als offener Leserahmen (*open reading frame* – ORF) wird eine lückenlose Folge von nicht-überlappenden Codons in der DNA-Sequenz definiert, die mit einem Startcodon beginnt und mit dem nächsten darauf folgenden Stoppcodon endet. Wenn mehrere Startcodons im gleichen Leserahmen liegen, ohne dass ein Stoppcodon im gleichen Leserahmen zwischen ihnen (*in-frame*) vorkommt, so werden diese alle als dem ORF zugehörig aufgefasst. Ein ORF, wie er hier definiert ist, umfasst also eine Menge von potentiellen Translationsstarts, die einem Stoppcodon zugeordnet sind, wobei nicht jeder ORF tatsächlich für ein Protein kodiert.

Zwischen den protein-kodierenden Sequenzenbereichen liegen solche, die nicht für Proteine kodieren (intergenische Regionen). Von diesen Bereichen wird angenommen, dass sie teilweise funktionslos sind. Sie können aber auch nicht translatierte RNA-Gene, RNA-Schalter (*riboswitches*) oder Bindestellen für die RNA-Polymerase, Ribosomen (RBS - *ribosome binding site*), Transkriptionsfaktoren und andere Signalmoleküle enthalten.

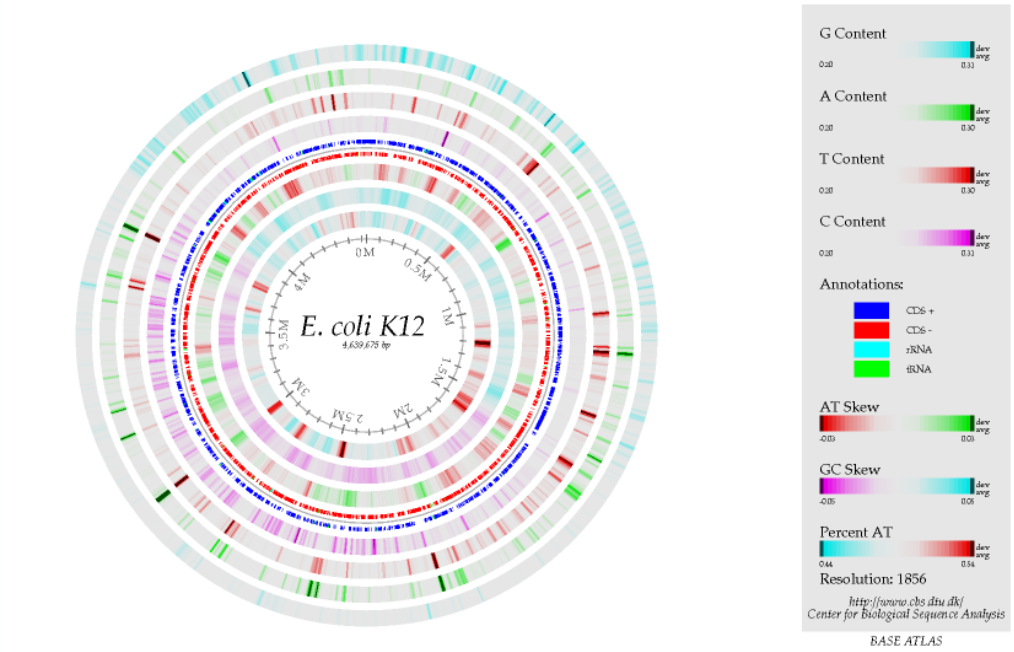

Abbildung 1.3: ***Genome Atlas Plot* des Genoms von *E. coli* K-12 [8]. Die Abbildung zeigt eine Übersicht der Annotation (CDS – *coding sequence*), sowie der Basenverteilungen des circulären Bakterienchromosoms.**

Eine Übersichtsdarstellung eines kompletten bakteriellen Genoms ist in Abbildung 1.3 gegeben. Die Abbildung zeigt exemplarisch die kodierenden Regionen (fünfter Kreis von außen, dunkelblau und dunkelrot), einige wichtige RNA-Gene, sowie den Gehalt der Nukleotide in verschiedenen Regionen des Genoms für den Modellorganismus *Escherichia coli* K-12.

1.2 Genvorhersage in prokaryotischen Genomen

Der erste Schritt bei der Genvorhersage in prokaryotischen Genomen ist bei allen gegenwärtigen Ansätzen die Suche nach ORFs einer gewissen Mindestlänge. Sie enthalten mit hoher Wahrscheinlichkeit kodierende Regionen. Komparative Ansätzen gleichen gefundene ORFs mit Datenbanken ab, in denen bereits annotierte Gene gespeichert sind. Diese Ansätze beruhen auf der Annahme, dass eine Nukleotidsequenz, die in einer Aminosäuresequenz übersetzt wird, welche hinreichende Ähnlichkeit zu einem bereits bekannten Protein aufweist, mit hoher Wahrscheinlichkeit kodierend ist [9]. Zwei gravierende Nachteile bei diesem Vorgehen sind offensichtlich: Zum einen können so nur Genen gefunden

werden, die zu bereits bekannten Genen hinreichend ähnlich sind, zum anderen ist ein Großteil annotierter Gene nicht experimentell belegt, so dass hier die Gefahr einer transitiven Fehlerfortpflanzung durch falsche Annotationen besteht.

Nicht-komparative Ansätze erstellen mit Hilfe der ORFs statistische Modelle, mit denen dann im gesamten Genom nach potentiell kodierenden Regionen gesucht wird. Dabei wird ausgenutzt, dass sich die Verteilung von Nukleotiden und Oligonukleotiden in kodierenden Regionen von der Verteilung in nicht-kodierenden Regionen signifikant unterscheidet. Das beruht darauf, dass Mutationen in kodierenden Regionen nicht zufällig verteilt sein können, da genetische Veränderungen nur dauerhaft im Genom bestehen, wenn sie entweder keine Funktionalität verändern (beispielsweise durch synonyme Codons) oder der Spezies durch die Veränderung einen Wettbewerbsvorteil verschaffen.

Neuere *ab initio* (nur auf der Eingabesequenz selbst basierend) Genvorhersageprogramme wie GLIMMER [10] oder GENEMARK [11] sagen bereits 98-99% der Gene eines Organismus korrekt vorher. Aber auch diese Ansätze haben Nachteile, zum einen ist die Zahl falsch-positiver Vorhersagen meist hoch, was beispielsweise durch Artefakte von Genen (Gene, die nicht exprimiert werden) im Genom verursacht werden kann, zum anderen sind die Modelle oft nicht »fein« genug, so dass die Vorhersage der Translationsstarts in vielen Fällen nicht korrekt ist. Ein weiteres Problem der automatisierten Genvorhersage besteht darin, dass prokaryotische Genome wie bereits beschrieben hinsichtlich der Charakteristika der kodierenden Regionen nicht immer *homogen* sind, so dass beispielsweise die Vorhersage in genomischen Inseln oft komplett versagt.

Die Falsch-Annotation von Translationsstarts ist teilweise darin begründet, dass viele der klassischen Genvorhersageprogramme die ORF-Längen maximieren. Eine andere Ursache für die ungenügende Genauigkeit ist möglicherweise, dass den kodierenden Regionen direkt nach dem Translationsstart andere Modelle zu Grunde gelegt werden müssen als anderen kodierenden Regionen. Systematische Studien belegen, dass der Fehler bei der Vorhersage der Translationsstarts insbesondere in Genomen mit hohem GC-Gehalt (GC-Gehalt über 60%) bei bis zu 30% liegen kann [12, 13, 14]. Die exakte Vorhersage des Translationsstarts ist aber von entscheidender Bedeutung, denn oft sind in der N-terminalen Region des Proteins funktionelle Signale kodiert. So kann diese Region Aufschluss geben über die Lokalisation des Proteins in der Zelle [15], über die Lebensdauer des Proteins [16] und die Rate mit der es exprimiert wird [17]. Außerdem ist es wichtig die exakten Koordinaten intergenischer Regionen zu kennen, da nur so das Auffinden von Promotoren, Bindestellen für Transkriptionsfaktoren und RNA-Genen möglich ist.

Generell bedarf die computergestützte Genvorhersage nach wie vor einer intensiven manuellen Nachbearbeitung durch Experten. Jede Verbesserung der automatischen Ver-

fahren bedeutet eine Verringerung des manuellen Aufwands, was besonders für Hochdurchsatz-Labore von großer Bedeutung ist.

1.2.1 Verfügbare Datenbanken

Unter den zahlreichen Anbietern freier und kommerzieller Datenbanken von Genomsequenzen, ist als wichtigster das National Center for Biotechnology Information (NCBI) [18] zu nennen. Hier sind alle bislang veröffentlichten Nukleotid- und Proteinsequenzen in der GenBank-Datenbank (GBK) verfügbar. Diese Datenbank ist allerdings in erster Linie auf Vollständigkeit ausgerichtet und nicht sehr gut gepflegt. Nur ein kleiner Teil der Daten ist verifiziert. Ein Großteil der Annotationen in GenBank stammen aus automatisierten Vorhersagen und ist seit der Erstveröffentlichung unverändert, selbst wenn es bereits überarbeitete Daten gibt. Hauptgrund hierfür ist die dezentrale Verantwortlichkeit für die Daten, die bis auf weiteres bei den Erstautoren bleibt. Andere Datenbanken wie beispielsweise SwissProt, des Swiss Institute of Bioinformatics (SIB) und European Bioinformatics Institute (EBI) [19] sind zentral verwaltet und gut gepflegt, enthalten aber nur einen Bruchteil der GenBank-Daten. Es ist daher schwierig verläßliche Annotationen für ein ganzes Genom zu finden, welches als vertrauenswürdige Referenz anerkannt wird.

Am nächsten kommen diesem Ziel der EcoGene-Datensatz [20], eine Zusammenstellung der überarbeiteten Annotationen des gut untersuchten Organismus *Escherichia coli* K-12 [21, 20]. Dieser Datensatz enthält über 800 experimentell verifizierte Gene und wird standardmässig als Referenzannotation bei der Evaluation von Genvorhersageprogrammen verwendet. Weitere manuell überarbeitete Datensätze sind unter anderem für die Organismen *Bacillus subtilis* [22] und *Pseudomonas aeruginosa* PAO1 [23, 24] verfügbar.

Kapitel 2

Klassifikation mit Methoden des Maschinellen Lernens

In diesem Kapitel werden einige Grundlagen des Maschinellen Lernens und ihrer Anwendung erläutert, welche zum Verständnis der verwendeten Algorithmen erforderlich sind.

2.1 Einleitung

In der vorliegenden Arbeit sollen Ansätze zur Verbesserung der Vorhersage von Translationsstarts untersucht werden. Wie im vorigen Kapitel beschrieben, können einem ORF (*open reading frame*) mehrere potentielle Translationsstarts zugeordnet werden, die im Folgenden auch als *Kandidaten* bezeichnet werden. Für jeden der Kandidaten ist zu entscheiden, ob es sich um einen *wahren* Translationsstart oder um einen *falschen* Translationsstart handelt. Als *falsche* Translationsstarts werden hier diejenigen Codons der Menge ATG, GTG, TTG bezeichnet, die im gleichen Leserahmen liegen wie das Gen, ohne dass ein Stoppcodon zwischen ihnen und dem annotierten Stopp auftritt, welche aber nicht für einen Translationsstart kodieren. Für jeden Kandidaten ist also eine binäre Entscheidung zu treffen. Zur Lösung dieses Problems sollen Ansätze des Maschinellen Lernens untersucht werden.

Mit Maschinellem Lernen können generell Methoden bezeichnet werden, mit denen statistische Zusammenhänge oder Gesetzmäßigkeiten anhand von Beispielen rekonstruiert werden. Gelernte Zusammenhänge können dann wiederum zur Vorhersage auf neuen Daten genutzt werden. Ausgehend von einer Reihe beobachteter Merkmale kann beispielsweise die Vorhersage von Klassenzugehörigkeiten von Beispielen gelernt werden.

In diesem konkreten Fall sollen die Charakteristika der wahren Translationsstarts und die der falschen Translationsstarts gelernt werden, mit dem Ziel für neue Kandidaten zuverlässige Vorhersagen treffen zu können. Merkmale oder Charakteristika sind in diesem Zusammenhang Sequenzmuster (*Motive*), d. h. um jeden potentiellen Translationsstart wird ein »Sequenzfenster« betrachtet, in welchem das Auftreten von Nukleotidmustern ausgewertet werden kann. Das Vorkommen von Nukleotiden oder Oligonukleotiden kann dabei für jede Kandidatensequenz positionsabhängig (Vorkommen an einer bestimmten Position der Sequenz) oder positionsunabhängig (Vorkommen irgendwo in der Sequenz) betrachtet werden.

2.2 Kodieren von Merkmalen

Um die Vorkommen von Nukleotidmustern als Merkmale für einen Lernalgorithmus zu verwenden, müssen sie in geeigneter Form beispielsweise als Vektoren dargestellt werden. Jedem Merkmal, d. h. jedem Vorkommen eines (Oligo-)Nukleotides, kann dann eine Dimension des Vektorraumes zugeordnet werden. Betrachtet man beispielsweise das positionsabhängige Vorkommen von Mononukleotiden in einer Sequenz der Länge L, so könnten diese folgendermaßen als Binärvektoren kodiert werden: Jede Position wird durch einen Vektor $\mathbf{z}_i \in \{0,1\}^4$ dargestellt, wobei jede Dimension des Vektors für das Vorkommen eines Nukleotides an der entsprechenden Position steht. Die erste Dimension könnte beispielsweise für Adenin stehen, die zweite für Cytosin usw. Das Vorkommen eines Nukleotides an der Position wird durch eine 1 kodiert, alle anderen Einträge des Vektors sind dann 0. Steht beispielsweise an der ersten Position der Sequenz ein Adenin so gilt $\mathbf{z}_1 = [1,0,0,0]^\top$. Die Vektoren der einzelnen Positionen der Sequenz können zu einem Vektor $\mathbf{x} \in \{0,1\}^d$, mit $d = 4L$ zusammengefasst werden:

$$\mathbf{x} = \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \\ \vdots \\ \mathbf{z}_L \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}.$$

Durch die Vektoren wird so ein d-dimensionaler Merkmalsraum aufgespannt.

Die Merkmalsvektoren von n Sequenzen können wiederum als Matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$ zusammengefasst werden, wobei mit $\mathbf{x}_i$ im Folgenden immer der $i$te Vektor der Matrix bezeichnet wird.

2.3 Klassifikation

Wahre Translationsstarts und falsche Translationsstarts können als zwei Kategorien betrachtet werden. Jeder Kandidat ist einer der Kategorien eindeutig zuzuordnen, was als *Klassifikation* bezeichnet wird. Allgemein wird mit Klassifikation die systematische Zuordnung von Objekten zu Klassen oder Kategorien bezeichnet. Die Zuordnung erfolgt anhand bestimmter Eigenschaften, welche die Elemente der verschiedenen Klassen unterscheiden. Ein Algorithmus, welcher die Klassifikation vornimmt, wird als *Klassifikator* bezeichnet. Die Eingabe eines solchen Algorithmus, sind die zu klassifizierenden Objekte, in diesem Fall die Vektoren der Oligonukleotidhäufigkeiten, die Ausgabe ist ein *Label*, welches der vorhergesagten Kategorie entspricht.

Ein wichtiger Aspekt bei der Klassifikation ist die Bewertung der Merkmale: In welchen Merkmalen unterscheiden sich die Elemente verschiedener Kategorien? Das Finden und die Gewichtung diskriminativer Merkmale kann durch einen Lernalgorithmus erfolgen. Grundsätzlich sind hier zwei Ansätze zu unterscheiden, das überwachte Lernen (*supervised learning*) und das unüberwachte Lernen (*unsupervised learning*). Beim überwachten Lernen wird von Beispielen mit bekanntem Label gelernt, während beim unüberwachten Lernen von den Beispielen ohne bekanntes Label gelernt wird. Auf beide Verfahren wird in den folgenden Abschnitten näher eingegangen.

Generell ist die Vorhersage nach dem Grundsatz des Maschinellen Lernens um so besser, je mehr repräsentative Beispiele beim Lernen verwendet werden. Dagegen wird die Vorhersage nicht unbedingt besser, wenn mehr *Merkmale* bei der Vorhersage einbezogen werden. Wenn die Merkmale keine relevanten Informationen für die Klassifikation tragen, ist es möglich, dass die Vorhersageperformanz dadurch nicht signifikant beeinflusst wird. In diesem Fall steigt lediglich der Rechenaufwand.

2.3.1 Überwachtes Lernen

Der Prozess des Lernens von Beispielen mit bekanntem Label wird allgemein als Training bezeichnet. Sei $\mathbf{x}_i \in \mathbb{R}^d$ ein Eingabevektor und $y_i \in \{1, \ldots, k\}$ das zugehörige Label, dann ist $(\mathbf{x}_i, y_i)$ ein Trainingsbeispiel. Die Menge der Beispiele mit bekanntem Label wird als Trainingsmenge $\mathcal{T} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ bezeichnet. Das Ziel des Trainings ist jedoch nicht auf Trainings-Eingabe/Ausgabe-Paaren 100% richtige Vorhersagen zu machen, sondern Eigenschaften der Verteilung explizit zu machen. Nur so kann eine gute Generalisierung erreicht werden, so dass mit dem gelernten Klassifikator auch für Beispiele, die nicht zum Training verwendet wurden, gültige Vorhersagen gemacht

werden können. Fehler bei der Vorhersage sind im Allgemeinen unvermeidbar (siehe *Bayes*-Risiko, Abschnitt 2.3.3, S. 15). Es gilt den Fehler zu minimieren, so dass eine Balance zwischen Generalisierung und Vorhersagegenauigkeit auf den Trainingsdaten erreicht wird. Wird diese Balance nicht erreicht, so spricht man von »Überanpassung« (*overfitting*), so dass das gelernte Modell nur die Merkmale der Beispiele aus dem Training beschreibt. Für ungesehene Eingaben, die den Trainingsdaten nicht exakt entsprechen, sind die Vorhersagen dann in den meisten Fällen fehlerhaft (siehe Abbildung 2.1).

Exemplarisch für ein überwachtes Lernverfahren wird in Abschnitt 2.5.1 (S. 20) die ›Methode der kleinsten Quadrate‹ (LSQ – *least squares method*) beschrieben.

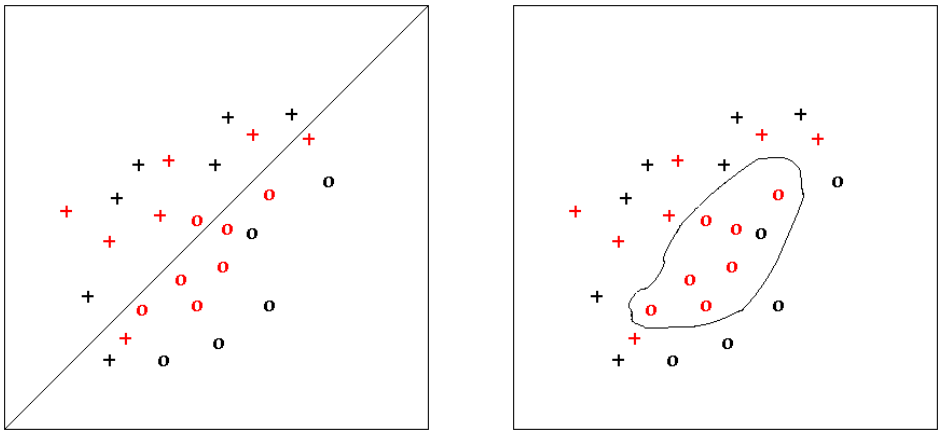

Abbildung 2.1: **Schematische Darstellung zur binären Klassifikation mit einem überwachten Verfahren am zweidimensionalen Beispiel. Die Trainingsbeispiele sind jeweils in rot dargestellt, die Testbeispiele in schwarz, Klasse +: Trainingsbeispiele +, Testbeispiele +; Klasse o: Trainingsbeispiele o, Testbeispiele o. Das Training besteht darin, die Parameter der Trennfunktion zu bestimmen, so dass sie die Beispiele der Klasse + und der Klasse o trennt. Linkes Bild: Die lineare Trennfunktion erreicht auf den Trainingsdaten nicht 100% Vorhersagegenauigkeit, zeigt aber einge gute Generalisierung, so dass die meisten Testdaten korrekt klassifiziert werden. Rechtes Bild: Mit der nicht-linearen Trennfunktion wird auf den Trainingsdaten eine Vorhersagegenauigkeit von 100% erreicht, ein Großteil der Testdaten wird aber falsch klassifiziert (schlechte Generalisierung).**

2.3.2 Unüberwachtes Lernen

Im Gegensatz zum überwachten Lernen gibt es beim unüberwachten Lernen (*unsupervised learning*) keine Beispieldaten mit vorher bekannter Klassenzugehörigkeit. Es gibt also kein direktes Maß für die Güte der Vorhersage. Diese muss in den meisten Fällen durch Experten beurteilt werden. Das Ziel bei unüberwachten Verfahren besteht darin,

relevante Strukturen in den Daten zu finden und sie entsprechend ihrer Ähnlichkeiten zu gruppieren. Oft werden die Gruppierungen mit iterativen Verfahren ermittelt.

Es gibt verschiedene Ansätze des unüberwachten Lernens, beispielsweise das Schätzen der Dichte von Verteilungen oder Verfahren zum Clustern von Daten. Das Clustern von Daten kann sowohl durch überwachte als auch durch unüberwachte Lernverfahren realisiert werden. Im Folgenden wird nur auf das unüberwachte Clustern von Daten eingegangen, da ein solches Verfahren in der vorliegenden Arbeit verwendet wird.

Die Gruppierung der Daten kann beispielsweise über die Distanz zwischen den Datenvektoren oder die Varianz innerhalb von Gruppen erfolgen. So kann ein Cluster als Gruppe von Punkten aufgefasst werden, die zueinander oder zu einem *Prototypen* (beispielsweise dem Schwerpunkt der Gruppe) eine geringere Distanz haben als zu den Punkten anderer Cluster. Das Maß für die Distanz kann dabei sehr unterschiedlich definiert sein, Beispiele sind der euklidische Abstand oder die City-Block-Metrik. Problematisch ist bei iterativen Clusterverfahren meist die Initialisierung und generell die Bestimmung der Anzahl von Clustern, denen die Datenpunkte zugeordnet werden sollen.

Ein bekanntes Beispiel für ein Clusterverfahren ist der prototypbasierte *K-Means*-Algorithmus, der in Abschnitt 2.5.2, S. 22 als Beispiel für ein unüberwachtes Verfahren beschrieben wird.

2.3.3 Klassifikationsrisiko

Betrachtet man die Klassifikation einer Menge von Datenpunkten, die gemäß einer statistisch unabhängigen Stichprobe realisiert wurde, so besteht bei jeder Vorhersage das Risko von Fehlklassifikationen. Diese werden mit »Kosten« belegt, welche durch eine geeignete Verlustfunktion festgelegt werden. Um das Risiko einer Fehlklassifikation zu minimieren, muss der Erwartungswert der Verlustfunktion minimal sein. Der *Bayes*-Klassifikator minimiert das Risiko einer Fehlklassifikation, indem die Zugehörigkeit zu einer Klasse durch die Dichtefunktionen der klassenspezifischen Merkmalsverteilungen ermittelt werden. Die Klassifikation erfolgt gemäß der größten Wahrscheinlichkeit der Klassenzugehörigkeit [25].

Sei $\mathbf{x} \in \mathbb{R}^d$ ein Merkmalsvektor im Sinne einer mehrdimensionalen Zufallsvariable[1] und C die Klassifikationsfunktion, die das Label $y_i \in Y$ vorhersagt. Die Wahrscheinlichkeitsdichte von $\mathbf{x}$ sei $p(\mathbf{x})$. Die klassenspezifische Wahrscheinlichkeitsdichte, von $\mathbf{x}$ für eine Klasse y_i wird durch $p(\mathbf{x}|y_i)$ angegeben. $P(y_i)$ sei die *a priori* Wahrscheinlichkeit, dass eine Eingabe zu einer Klasse $y_i \in Y$ gehört.

[1] Im Folgenden wird in der Notation nicht zwischen Zufallsvariablen und Realisierungen (Datenpunkten) unterschieden.

Für die Verbunddichte von Merkmalsvektor und Label gilt:

$$p(\mathbf{x}, y_i) = p(\mathbf{x}|y_i)P(y_i) = P(y_i|\mathbf{x})p(\mathbf{x}). \tag{2.1}$$

Nach dem Bayes-Theorem erhält man die *a posteriori* Wahrscheinlichkeit, dass ein Eingabevektor zur Klasse y_i gehört:

$$P(y_i|\mathbf{x}) = \frac{p(\mathbf{x}|y_i)P(y_i)}{p(\mathbf{x})}. \tag{2.2}$$

Der zugehörige Bayes-Klassifikator:

$$C(\mathbf{x}) = \arg\max_{y_i} P(y_i|\mathbf{x}) \tag{2.3}$$

minimiert den erwarteten Fehler bei der Klassifikation bei gleichen Kosten für alle Arten von Fehlklassifikation.

Dieser minimale Fehler (auch als *Bayes*-Risiko bezeichnet), wird in den meisten realen Fällen nicht erreicht, da die bedingten Wahrscheinlichkeiten sich nicht genau bestimmen lassen. Die entsprechenden Wahrscheinlichkeit bzw. Dichten müssen aus einer Stichprobe geschätzt werden, wobei die Güte dieser Schätzung ganz wesentlich vom Umfang der Stichprobe, von der Dimensionalität der Daten und davon wie gut das Verteilungsmodell die Realität widerspiegelt, abhängt.

2.3.4 Lineare Klassifikation

Ein gut untersuchter Klassifikationsansatz besteht in der linearen Trennung der Daten mittels einer sogenannten lineare Diskriminante. Diese kann man sich geometrisch betrachtet als räumliche Trennung der Datenpunkte durch Hyperebenen vorstellen, d. h. für eine binäre Klassifikation wird der Datenraum in zwei Halbräume geteilt. Im Idealfall enthält der eine Halbraum alle Punkte einer Klasse, der andere alle der anderen. Ist dies der Fall, so spricht man von linearer Separabilität der Daten. Für eine Eingabe kann mit Hilfe der Trennfunktion bestimmt werden, welchem Halbraum und damit welcher Klasse sie zugeordnet wird.

Für Eingabedaten $\mathbf{x} \in \mathbb{R}^d$ kann eine lineare Trennfunktion durch

$$\begin{aligned} f &: \mathbb{R}^d \mapsto \mathbb{R} \\ f(\mathbf{x}) &= \langle \mathbf{w}, \mathbf{x} \rangle + b \end{aligned} \tag{2.4}$$

repräsentiert werden. Der Ausdruck $\langle \mathbf{w}, \mathbf{x} \rangle$ ist definiert als *inneres Produkt* (Skalarprodukt) zwischen dem Richtungsvektor der Hyperebene $\mathbf{w}$ und dem Datenvektor $\mathbf{x}$. Die

Lage der Ebene relativ zum Ursprung wird durch b spezifiziert. Die Entscheidungsregel (Klassifikator) $C : \mathbb{R}^d \rightarrow \{-1,\ 1\}$ für einen Datenpunkt $\mathbf{x}$ lautet:

$$C(\mathbf{x}) = \begin{cases} 1, \text{falls } f(\mathbf{x}) > 0 \\ -1, \text{sonst.} \end{cases} \tag{2.5}$$

2.3.5 Nicht-lineare Klassifikation

Generell würde man erwarten, dass die Vorhersagegenauigkeit auf den Trainingsdaten um so besser ist, desto komplexer die Menge zugelassener Funktionen ist. Eine lineare Trennfunktion hat weniger Freiheitsgrade und ist somit weniger flexibel als nicht-lineare Funktionen. Komplexere Funktionsklassen können in vielen Fällen die Trainingsdaten besser abbilden, das Risiko einer Überanpassung (*overfitting*) ist im Allgemeinen jedoch größer (siehe Abbildung 2.1, S. 14).

Da es oft keine Hyperebene gibt, welche die Klassen im Eingaberaum hinreichend gut trennt, lassen sich viele Probleme nicht optimal durch einen linearen Klassifikator lösen. Es gibt verschiedene Verfahren, um eine nicht-lineare Klassifikation zu erreichen, beispielsweise neuronale Netze [26] oder kernbasierte Verfahren [27]. Bei kernbasierten Verfahren umgeht man das Problem, eine nicht-lineare Trennfunktion zu schätzen: Die Daten werden mittels einer nicht-linearen Abbildungsfunktion in einen Merkmalsraum (*feature space*) transformiert, um dort eine lineare Trennung durchzuführen. Der in dieser Arbeit vorgestellte Oligo-Kern-Algorithmus ist ein solches kernbasiertes Verfahren, daher werden im folgenden Abschnitt die Grundlagen kernbasierten Lernens beschrieben.

2.4 Kernbasiertes Lernen

In jüngster Vergangenheit wurden zahlreiche Verfahren des Maschinellen Lernens auf bioinformatische Problemstellungen angewendet. Vor allem Verfahren wie Hidden Markow-Modelle (HMM) [28], neuronale Netze (NN) [29] und Support Vektor Maschinen (SVM) [27] werden bei der Modellierung und Analyse von biologischen Sequenzdaten eingesetzt. Klassische Markow-Modelle und neuronale Netze haben den Nachteil, dass sie ohne Regularisierung in hochdimensionalen Räumen eine große Menge an Trainingsdaten benötigen, welche für viele biologische Problemstellungen nicht verfügbar sind. Außerdem ist die Parameteroptimierung bei diesen Verfahren vergleichsweise schwierig. Anders ist dies bei kernbasierten Methoden, bei denen die Optimierung durch lineare oder quadratische Programmierung bzw. Lösen eines linearen Gleichungssystems reali-

siert werden kann. Die Anzahl der freien Parameter ist nicht abhängig von der Anzahl der Dimensionen und durch gut untersuchte Regularisierungsmechanismen kann mit kernbasierten Methoden auch in hochdimensionalen Merkmalsräumen mit wenigen Beispielen gelernt werden [27].

2.4.1 Prinzip des kernbasierten Lernens

Das Ziel kernbasierten Lernens ist, eine nicht-lineare Trennfunktion mit Hilfe des sogenannten »Kern-Tricks« zu finden, der im Folgenden erläutert wird.

Für viele Klassifikationsprobleme ist eine lineare Trennung im Eingaberaum $\mathcal{X}$ (*input space*) nicht optimal. Daher bildet man die Daten mit einer geeigneten Funktion $\phi(\mathbf{x})$ in einen Merkmalsraum $\mathcal{F}$ (*feature space*) ab, in dem eine lineare Trennung erfolgt. $\phi(\mathbf{x})$ ist im Allgemeinen eine nicht-linearen Abbildung (Transformation) des Eingaberaumes $\mathcal{X}$ auf den Merkmalsraum $\mathcal{F}$. Eine schematische Darstellung der Transformation von Daten aus dem Eingaberaum in einen Merkmalsraum ist in Abbildung 2.2 gegeben.

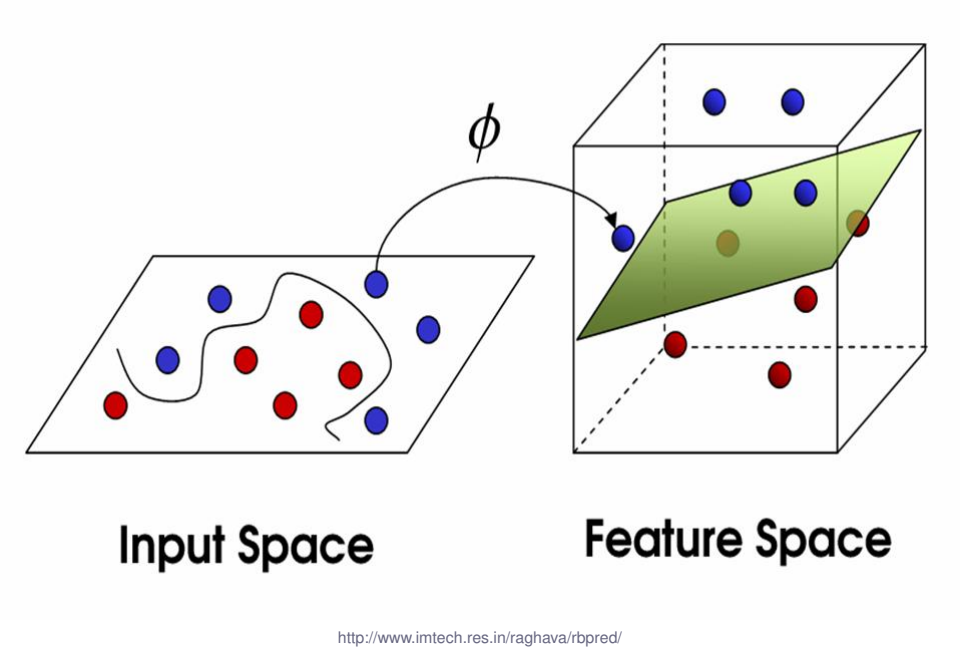

Abbildung 2.2: **Schematische Darstellung einer nicht-linearen Abbildung ϕ der Datenpunkte aus dem Eingaberaum (*Input Space*) in einen Merkmalsraum (*Feature Space*). Im Eingaberaum liefert ein lineare Trennung der Daten kein zufriedenstellendes Ergebnis. Durch eine geeignete Abbildung in einen Merkmalsraum, kann eine sinnvolle nicht-lineare Trennung im Eingaberaum mit einer linearen Diskriminanten im Merkmalsraum erreicht werden.**

Diskriminante im Merkmalsraum:

Die Konstruktion eines binären linearen Klassifikators im Merkmalsraum $\mathcal{F}$ erfordert einen Gewichtsvektor $\mathbf{w} \in \mathcal{F}$ und einen Bezug zum Ursprung, der im Folgenden mit b bezeichnet wird. Die gesuchte Trennfunktion hat die Form

$$f(\mathbf{x}) = \langle \mathbf{w},\ \phi(\mathbf{x}) \rangle + b. \tag{2.6}$$

In dieser Form wird die Diskriminante *explizit* berechnet, d. h. sie muss für jeden Datenpunkt mittels der Abbildung $\phi(\mathbf{x})$ berechnet werden. Mit Hilfe des Vorzeichens erfolgt die Zuordnung eines Elementes zu einer der beiden Klassen analog zu (2.5). Das Berechnen der Diskriminante ist um so aufwendiger, je komplexer die Abbildung in den Merkmalsraum ist.

Kernbasierte Repräsentation

Eine wichtige Eigenschaft linearer Klassifikatoren ist, dass sie in eine *kernbasierte* Form überführt werden können. Durch die kernbasierte Repräsentation ist eine Klassifikation der Daten im Merkmalsraum möglich, ohne dass die Diskriminante im Merkmalsraum explizit berechnet werden muss. Dazu wird lediglich das *innere Produkt* von Datenpaaren betrachtet, was als »Kern-Trick« bezeichnet wird.

Für einen Eingaberaum $\mathcal{X}$ ist ein Kern eine Funktion, so dass für alle $\mathbf{x},\ \mathbf{x}' \in \mathcal{X}$ gilt

$$k(\mathbf{x},\ \mathbf{x}') = \langle \phi(\mathbf{x}),\ \phi(\mathbf{x}') \rangle \tag{2.7}$$

Das innere Produkt der transformierten Merkmale $\phi(\mathbf{x})$ und $\phi(\mathbf{x}')$ kann als Maß für die Ähnlichkeit zweier Elemente $\mathbf{x},\ \mathbf{x}' \in \mathcal{X}$ aufgefasst werden. Für die Klassifikation werden die inneren Produkte zwischen den zu klassifizierenden Daten und den Trainingsdaten unter Einbeziehung einer gelernten Gewichtung berechnet. Die Trennfunktion ist somit eine lineare Kombination von Kernfunktionen aller Trainingsdaten $\mathbf{x}_i$ mit $i \in \{1, \ldots, n\}$ und der zu klassifizierenden Objekte, gewichtet durch einen Parameter α_i. Für ein zu klassifizierendes Objekt $\mathbf{x}$ erhält man die Diskriminante (2.6) in der kernbasierten Form:

$$f(\mathbf{x}) = \phi(\mathbf{x}) \cdot \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x},\ \mathbf{x}_i), \tag{2.8}$$

wobei b weggelassen werden kann, wenn die Trennebene durch den Ursprung gelegt wird. Die Gewichtung α_i wird durch das Training mit gelabelten Bespielen gelernt. Sie

legt fest wie stark ein Element der Trainingsmenge in die Diskriminante eingeht. Die Zuordnung zu einer der beiden Klassen erfolgt entsprechend (2.5) durch das Vorzeichen von $f(\mathbf{x})$.

Die Gesamtheit der inneren Produkte kann in der Kernmatrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ mit Elementen

$$\mathbf{K}_{ij} = k(\mathbf{x}_i,\ \mathbf{x}_j), \forall\ i,\ j \in \{1,\ 2,\ \ldots,\ n\} \tag{2.9}$$

zusammengefasst werden. Die Gewichte werden im Vektor $\boldsymbol{\alpha}$ zusammengefasst. Der Vorteil kernbasierter Verfahren liegt darin, dass die Anzahl der freien Parameter in der kernbasierten Form nicht von der Anzahl der Eingabe-Dimensionen abhängt. So kann auch eine lineare Trennung in abstrakten Merkmalsräumen (beispielsweise Funktionsräumen) gelernt werden, obwohl die Diskriminante hier nicht mehr explizit dargestellt werden kann. Zudem wird der Aufwand bei der Berechnung im Allgemeinen nicht durch die Anzahl der Dimensionen des Merkmalsraumes beeinflusst. Für die Klassifikation eines Datenpunktes der Testmenge werden maximal so viele innere Produkte berechnet, wie Beispiele beim Training verwendet werden.

2.5 Grundlagen verwendeter Algorithmen

Im Folgenden werden grundlegend die ›Methode der kleinsten Quadrate‹ und der K-Means-Algorithmus als Beispiele für Lernverfahren vorgestellt. Beide Verfahren sind mit den im Rahmen dieser Arbeit entwickelten Verfahren verwandt.

2.5.1 Methode der kleinsten Quadrate

Als Beispiel für ein überwachtes Lernverfahren wird die ›Methode der kleinsten Quadrate‹ [30] (LSQ – *least squares method*) vorgestellt. Die regularisierte Variante RLSQ (*regularized least squares*) [31] ist ein Verfahren mit dem eine lineare Diskriminante in allgemeinen Merkmalsräumen (siehe 2.4) geschätzt werden kann. Dieses Verfahren wird für das Training bei dem in dieser Arbeit vorgestellten Oligo-Kern-Algorithmus verwendet.

Zur Vereinfachung wird hier die Trennebene wieder durch den Ursprung gelegt:

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle. \tag{2.10}$$

Der Gradient $\nabla f(\mathbf{x}) = \mathbf{w}$ definiert die Ausrichtung der Ebene. Der optimale Vektor $\widehat{\mathbf{w}}$ wird geschätzt, indem die Summe der quadratischen Abweichung zwischen Label-Vektor und innerem Produkt von Gewichts- und Datenvektor minimiert wird:

$$\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{i=1}^{n} (y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle)^2. \tag{2.11}$$

Die transponierten Vektoren der Eingabedaten lassen sich in einer Matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ zusammenfassen. Die zugehörigen Label werden in einem Vektor $\mathbf{y}$ zusammengefasst. Durch Umschreiben der Summe als Matrix-Vektor-Produkt kann man die zu minimierende Fehlerfunktion wie folgt darstellen:

$$E(\mathbf{w}) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2. \tag{2.12}$$

Für ein Minimum muss der Gradient des Fehlers zum Nullvektor werden, d. h. es muss gelten:

$$\mathbf{X}^\top \mathbf{X}\mathbf{w} = \mathbf{X}^\top \mathbf{y}. \tag{2.13}$$

Wenn $\mathbf{X}^\top \mathbf{X}$ nicht-singulär (invertierbar) ist, gibt es eine eindeutige Lösung:

$$\widehat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}. \tag{2.14}$$

Für höherdimensionale Datenräume ist $\mathbf{X}^\top \mathbf{X}$ nicht invertierbar, daher wird in diesem Fall die regularisierte Variante angewendet:

$$E(\mathbf{w}) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda \|\mathbf{w}\|^2. \tag{2.15}$$

Für die Regularisierung wird ein Strafterm eingeführt, welcher der quadrierten Norm des Gewichtsvektors entspricht. Dieser Strafterm wird gewichtet durch den Regularisierungsparameter $\lambda > 0$ und zu dem Fehler bei der Klassifikation addiert. Die Norm des Gewichtsvektors wird dabei einbezogen, damit einzelne Dimensionen der Merkmalsvektoren nicht beliebig hoch gewichtet werden können. Der optimale Gewichtsvektor wird durch

$$\widehat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{y} \tag{2.16}$$

berechnet, wobei $\mathbf{I}$ die Identitätsmatrix ist.

2.5.2 K-Means-Algorithmus

Als Beispiel für unüberwachtes Lernen wird hier der K-Means-Algorithmus [32, 33] vorgestellt. Der K-Means-Algorithmus ist ein iteratives Clusterverfahren, bei dem die Varianz innerhalb der Cluster minimiert wird. Die Anzahl der Cluster K muss dem Algorithmus vorgegeben werden. Jeder Cluster wird durch einen Stellvertreter, den sogenannten *Prototypen* repräsentiert. Dieser kann beispielsweise als Schwerpunkt der Datenpunkte eines Clusters definiert sein. Es erfolgt zunächst die Initialisierung von K Prototypen. Eine mögliche Initialisierung ist eine zufällige Auswahl von K Datenvektoren als initiale Prototypen. Alle Eingabevektoren werden dann demjenigen Prototypen zugeordnet, dem sie am nächsten sind. Dabei kann z. B. der quadratische euklidische Abstand als Distanzmaß für einen Datenvektor $\mathbf{x}$ und Prototyp $\mathbf{m}$ verwendet werden:

$$D(\mathbf{x}, \mathbf{m}) \quad = \quad \|\mathbf{x} - \mathbf{m}\|^2. \tag{2.17}$$

Nach der Zuordnung der Datenpunkte zu den Prototypen, werden die Prototypen neuberechnet. Anschließend werden die Datenvektoren wieder den ihnen am nächstenliegenden Prototypen zugeordnet. Die beiden Schritte (Zuordnung und Neuberechnung der Prototypen) werden solange iteriert, bis sich die Zuordnung der Datenvektoren zu Clustern nicht mehr verändert.

Mit obiger Distanz (2.17) ist das Ziel die Minimierung der Verlustfunktion

$$E = \sum_{i=1}^{n} \sum_{j=1}^{K} h_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|^2, \tag{2.18}$$

mit

$$h_{ij} = \begin{cases} 1, \text{ falls } \mathbf{x}_i \text{ zu Cluster } j \text{ gehört} \\ 0, \text{ sonst.} \end{cases}$$

Eine Verallgemeinerung des K-Means-Algorithmus ist der EM-Algorithmus (*Expectation Maximization*) [34]. Der EM-Algorithmus gehört zu den Clusterverfahren mit »weicher« Zuordnung, d. h. ein Datenvektor wird nicht »hart« einem Cluster zugeordnet, sondern wird jedem Cluster mit einer gewissen Wahrscheinlichkeit zugeordnet. Die Prototypen sind beim EM-Algorithmus Cluster-spezifische Verteilungen deren Parameter mitgeschätzt werden müssen.

Der Algorithmus lässt sich in zwei Schritten zusammenfassen, die wie beim K-Means-Algorithmus iteriert werden:

1. Schätzen für jeden Datenvektor mit welcher Wahrscheinlichkeit er zu jedem der Cluster gehört.

2. Neubestimmen der Verteilungsparameter auf Grundlage der aktuellen Zuordnungswerte.

Der in dieser Arbeit vorgestellte Algorithmus für die unüberwachte Klassifikation von TIS-Kandidaten (implementiert in dem Tool TICO) kann als Spezialfall des EM-Algorithmus betrachtet werden.

Kapitel 3

Datamining auf prokaryotischen Genomsequenzen mit einem überwachten Lernverfahren

3.1 Lernalgorithmen für prokaryotische Translationsstarts

In diesem Teil der Arbeit soll zunächst mit einem überwachten Verfahren, dem Oligo-Kern-Algorithmus [1], die »Klassifizierbarkeit« prokaryotischer Translationsstarts an einem Datensatz mit experimentell belegter Annotation untersucht werden. Dabei soll auch die Anwendbarkeit des Algorithmus als Verfahren zum Datamining auf biologischen Sequenzen gezeigt werden. Es ist davon auszugehen, dass überwachte Verfahren eine bessere Vorhersageperformanz erreichen als unüberwachte, sofern ein repräsentativer Datensatz für das Training verwendet werden kann. Da für die meisten prokaryotischen Genome, insbesondere neu-sequenzierte Genome, jedoch keine verifizierte Annotation verfügbar ist, soll im zweiten Teil die Anwendbarkeit eines unüberwachten Verfahrens untersucht werden, mit dem Ziel ein performantes, in der Praxis der Genomannotation anwendbares Werkzeug zu entwickeln.

Als Fallstudie wurden zur Evaluation des Oligo-Kern-Algorithmus die Sequenzen experimentell verifizierter Translationsstarts (TIS – *translation initiation site*) des Eubakteriums *Escherichia coli* K-12 [21] untersucht. Eine detaillierte Beschreibung der verwendeten Daten ist in Abschnitt 3.3.1 (S. 33) gegeben. Bei der binären Klassifikation sollen *wahre* Translationsstarts von *falschen* Translationsstarts (TIS-Kandidaten, die nicht für einen Translationsstart kodieren) unterschieden werden. Vorrangiges Ziel war die Untersuchung der Anwendbarkeit des Oligo-Kern-Algorithmus zur Detektion von Signalen mit

biologischer Relevanz. Dabei wird die Hypothese zugrunde gelegt, dass von dem Klassifikator gelernte Merkmale, auch *in vivo* beim Erkennen einer TIS eine Rolle spielen. Um dies zu belegen, werden die gelernten Merkmale mit bekannten Mustern (*Motiven*) der Translationsinitiation bei *E. coli* verglichen.

Für die Vorhersage eukaryotischer Translationsstarts werden bereits seit 1997 erfolgreich Methoden des Maschinellen Lernens angewendet, Beispiele sind Pedersen *et al.*, die Neuronale Netze [35] verwenden, sowie Zien *et al.* [36] und Li und Jiang [37], die Support Vektor Maschinen in Kombination mit Kernen einsetzen. Zien *et al.* verwendet dabei einen modifizierten Polynomkern, während Li und Jiang sogenannte *Edit-Kerne* verwenden. Die Ansätze gehen jedoch davon aus, dass eine große Menge gelabelter Daten vorliegt. Da im Falle eukaryotischer Translationsstarts die Signale auch speziesübergreifend gelernt werden können, steht hier eine deutlich größere Menge verifizierter Trainingsdaten zur Verfügung als bei der Vorhersage prokaryotischer Genstarts, wo man davon ausgeht, dass viele Signale nicht speziesübergreifend sind.

Bei der Entwicklung der vorgestellten Methoden und ihre Evaluation gehen folgende Annahmen ein: 1. Relevante Informationen bezogen auf prokaryotische Translationsstarts sind in der unmittelbaren Umgebung der Translationsstarts lokalisiert; 2. die Positionsinformation ist von entscheidender Bedeutung; 3. sowohl der upstream-Bereich eines Translationsstarts als auch der downstream-Bereich ist informativ.

3.2 Oligo-Kern-Algorithmus

Ein Nachteil kernbasierter Verfahren ist in vielen Fällen ein Mangel an Transparenz für den Benutzer, so dass eine Bewertung und Analyse der Merkmale, die für die Klassifikation relevant sind, schwierig ist. Um die gelernten Charakteristika zu identifizieren, wurden daher Verfahren zur Selektion von Merkmalen entwickelt, die mit den klassischen kernbasierten Verfahren kombiniert werden können [38]. Im Gegensatz zu klassischen kernbasierten Lernverfahren ist bei dem in dieser Arbeit vorgestellten Oligo-Kern-Algorithmus eine intuitive Darstellung der gelernten Charakteristika ohne weiteres möglich. Die meisten kernbasierten Ansätze nutzen nur den Vorteil der kernbasierten Repräsentation, ohne die explizite Form der Diskriminante im Merkmalsraum zu berücksichtigen. Beim Oligo-Kern-Algorithmus werden beide Formen einbezogen: Die kernbasierte Form (*Oligo-Kerne*) ermöglicht das Lernen in hochdimensionalen Räumen mit verhältnismässig geringem Rechenaufwand, die Repräsentation der Diskriminanten im Merkmalsraum (*Oligo-Funktionen*) erlaubt die Interpretation der gelernten Merkmale.

Der Oligo-Kern-Algorithmus bietet einen weiteren Vorzug bei der Analyse biologi-

scher Sequenzen gegenüber bereits existierenden Methoden, die entweder auf vollständig positionsabhängigen Mononukleotid-Vorkommen [38, 36] oder positionsunabhängigen Oligomer-Vorkommen [39, 40] basieren. Beim Oligo-Kern-Algorithmus kann der Grad der Positionsabhängigkeit (Glättung) von Oligomer-Vorkommen mit Hilfe eines Glättungsparameters eingestellt werden. Damit ist der Oligo-Kern-Algorithmus eine Verallgemeinerung voriger Ansätze, da die Analyse von vollständig positionsabhängigen und positionsunabhängigen Oligomer-Vorkommen Spezialfälle sind, bei denen die Glättung gegen 0 bzw. ∞ geht. Durch die Anpassung des Grades an Positionsabhängigkeit, können komplexe biologische Signale, die häufig eine gewisse Variabilität in Position und Zusammensetzung aufweisen, gut modelliert werden. Ein alternativer Ansatz bei dem der Grad der Positionsabhängigkeit eingestellt werden kann, ist der 2004 von Li und Jiang vorgestellte Edit-Kern [37], bei dem die Ähnlichkeit von Sequenzen durch die *String-Edit-Distanz* (Levenshtein-Distanz) definiert wird.

Mit Hilfe einer Visualisierungsfunktion wird eine intuitive und transparente Darstellung der charakteristischen Sequenzmerkmale erreicht. Gelernte Merkmale werden mit ihrer Gewichtung bei der Klassifikation dargestellt. Da die Gewichtung eines Merkmals dessen Bedeutung für die Klassifikation widerspiegelt, kann der Benutzer so relevante Signale lokalisieren und identifizieren. Dies erlaubt wiederum Rückschlüsse für eine biologische Interpretation der Signale. Durch die Transparenz hat der Benutzer außerdem eine Möglichkeit zur Bewertung der Klassifikation und der Qualität der Eingabedaten.

3.2.1 Oligo-Funktionen und Glättung der Positionsinformation

Zur Glättung der Positionsinformation werden die diskreten positionsabhängigen Vorkommen der Oligomere (Kmere) in einer Sequenz als kontinuierliche Funktionen abgebildet. Dadurch kann ein Kmer-Vorkommen für jede Position der Sequenz als reeller Wert im Intervall $[0,1]$ angegeben werden. Die Glättung erfolgt mit Hilfe einer Gauss-Funktion, wobei der Grad der Unschärfe durch die Varianz σ eingestellt wird. Die geglätteten Funktionen der Oligomer-Vorkommen werden im Folgenden als *Oligo-Funktionen* bezeichnet.

Für eine Sequenz, in der ein Wort (Oligomer) $\omega \in \mathcal{A}^K$ an den Positionen $T_\omega = \{p_1, p_2, \dots\}$ vorkommt, wird die Oligo-Funktion μ über alle Positionen $t \in \mathbb{R}$ definiert als:

$$\mu_\omega(t) = \sum_{p \in T_\omega} e^{\left(-\frac{1}{2\sigma^2}(t-p)^2\right)}. \tag{3.1}$$

Das Vorkommen eines Kmers wird somit nicht »hart« den tatsächlichen Positionen zugeordnet, sondern hat Einfluss auf die jeweils benachbarten Positionen. Das Maximum

der Funktion liegt an der tatsächlichen Position des Vorkommens. Die Breite der Gauss-Glocke wird durch den Parameter σ eingestellt. Die Funktionen eines Oligomers werden jeweils aufaddiert (siehe Abbildung 3.1, S.28). Für $\sigma \mapsto \infty$ geht jegliche Positionsinformation verloren und die Vorkommen werden wieder positionsunabhängig betrachtet, für $\sigma \mapsto 0$ hat jedes Vorkommen nur Einfluss auf die tatsächliche Position, so dass sich benachbarte Vorkommen nicht beeinflussen.

Für eine DNA-Sequenz $s \in \mathcal{A}^L = \{A,\ C,\ G,\ T\}^L$ können die Vorkommen aller Kmere $\omega \in \{\omega_1, \omega_2, \ldots, \omega_{4^K}\}$ durch einen Vektor $\mathbf{x}$ repräsentiert werden. Der Merkmalsraum ist in diesem Fall der 4^K-dimensionalen Raum aller Kmer-Funktionen. Die Abbildung in den Merkmalsraum mit Hilfe der Oligo-Funktionen wird definiert als

$$\phi(\mathbf{x}) = [\mu_{\omega_1},\ \mu_{\omega_2},\ \ldots,\ \mu_{\omega_{4^K}}]^\top. \tag{3.2}$$

wobei die Funktion $\phi(\mathbf{x})$ vektorwertig ist. Für alle Positionen t wird die Funktion entsprechend angegeben als:

$$\boldsymbol{\phi}(t) = [\mu_{\omega 1}(t),\ \mu_{\omega 2}(t),\ \ldots\ \mu_{\omega_{4^K}}(t)]^\top.$$

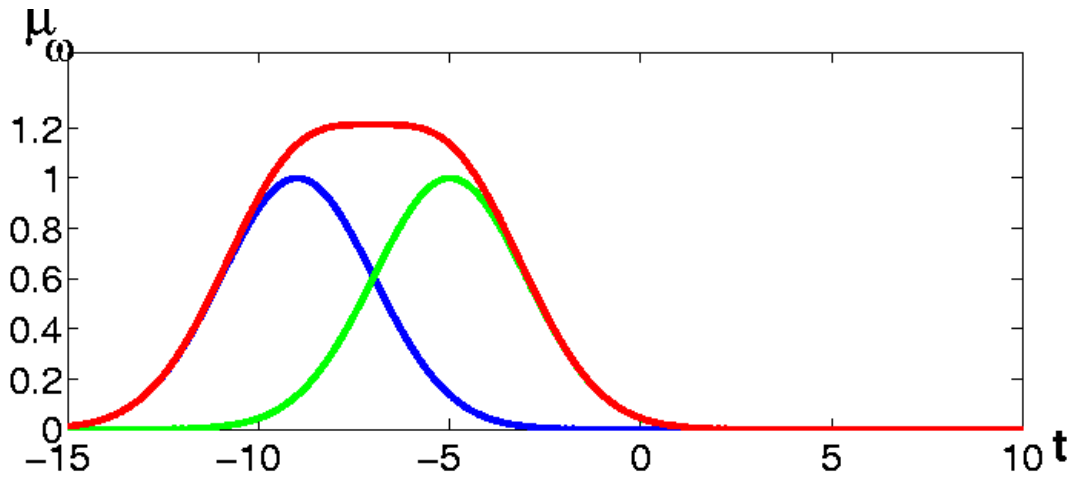

Abbildung 3.1: **Glättung der Positionsinformation einer Oligo-Funktion μ_ω (rote Kurve) mit Hilfe einer Gauss-Faltung. Im Beispiel wird ein Oligomer ω an den Positionen -9 (blaue Kurve) und -5 (grüne Kurve) beobachtet. Der Grad der Glättung hängt von der Varianz σ ab, je größer der Wert von σ ist, desto breiter sind die Gauss-Glocken, d. h. desto mehr wird die Positionsinformation »verschmiert«.**

3.2.2 Visualisierung der Oligo-Funktionen

Die Oligo-Funktionen selbst sind im Allgemeinen auf Grund des hohen Rechenaufwandes zum Lernen einer Diskriminante im Merkmalsraum ungeeignet (siehe dazu Abschnitt

2.4.1, S. 19). Sie erlauben aber in der diskretisierten Form eine Interpretation der Relevanz eines Merkmals durch eine Gewichtung mit gelernten Parametern α_i. Die resultierende Gewichtsfunktion $\mathbf{w}(t)$ aller Oligo-Funktionen über die Positionen t ist für n Trainingssequenzen definiert als

$$\mathbf{w}(t) = \sum_{i=1}^{n} \alpha_i [\mu^i_{\omega_1}(t),\ \mu^i_{\omega_2}(t),\ \ldots,\ \mu^i_{\omega_{4^K}}(t)]^\top. \tag{3.3}$$

Sie hält somit Informationen über den »Beitrag« der Oligomere an den Positionen t zur Klassifikation. Hohe positive Werte geben einen starken Hinweis für die Zuordnung zur positiven Klasse, während hohe negative Werte stark in die Klassifikation zur negativen Klasse eingehen.

Betrachtet man die Positionen in der Sequenz wieder als diskrete Werte t_j, so sind in dem Vektor $\mathbf{w}(t_1) \in \mathbb{R}^{4^K}$ die Gewichte der Kmer-Vorkommen an Position t_1 der Sequenz eingetragen. Die Gewichtsvektoren für alle Sequenzpositionen t_j können als Matrix $\mathbf{W} \in \mathbb{R}^{4^K \times L}$ zusammengefasst werden:

$$\mathbf{W} = [\mathbf{w}(t_1),\ \mathbf{w}(t_2),\ \ldots,\ \mathbf{w}(t_L)]. \tag{3.4}$$

Diese kann für kurze Oligomere ($K < 5$) komplett abgebildet werden (siehe dazu Ergebnisteil, Abbildung 3.3, S. 38). Für $K \geq 5$ wird die Darstellung aller Gewichte unübersichtlich, die Betrachtung ist dann nur noch ausschnittsweise sinnvoll.

Um den positionsabhängigen Beitrag einzelner Kmere zu analysieren, ist es möglich deren Gewichtsfunktion über alle Positionen der betrachteten Sequenzen abzubilden (siehe Ergebnisteil Abbildung 3.4, S. 39). Die Gewichtsfunktionen werden entsprechend ihrer L_2-Norm

$$N_i = \sqrt{\int w_i(t)^2 dt}, \quad i = 1,\ \ldots,\ 4^K \tag{3.5}$$

nach ihrer »Wichtigkeit« für die Diskriminanten geordnet. Alle Visualisierungen für die untersuchten Oligomere ($K \in \{1, 2, \ldots, 6\}$) am Beispiel *E. coli* K-12 sind unter http://gobics.de/oligo_functions verfügbar.

3.2.3 Oligo-Kerne: Kernbasierte Repräsentation der Diskriminante

Wie in Abschnitt 3.2.1 (S. 27f) beschrieben, repräsentiert die Oligo-Funktion $\phi(\mathbf{x})$ die vollständige Abbildung der Sequenz $\mathbf{x}$ im Merkmalsraum. Da diese aber für das Lernen einer expliziten Diskriminante ungeeignet ist, wird im Folgenden die kernbasierte Repräsentation mittels der sogenannten *Oligo-Kerne* eingeführt. Zur Vereinfachung der Notation

wird $\phi(\mathbf{x}_i) \equiv \phi_i$ gesetzt. Mit dem Kern-Trick (siehe Abschnitt 2.4.1, S. 19) kann die Diskriminante durch das innere Produkt zwischen Eingabesequenzen berechnet werden. Für die Repräsentation zweier Eingabesequenzen $\mathbf{x}_i$ und $\mathbf{x}_j$ ist das innere Produkt der Abbildungen definiert als

$$\langle \phi_i,\ \phi_j \rangle = \int \langle \phi_i(t),\ \phi_j(t) \rangle dt = \sum_{\omega \in \mathcal{A}^K} \sum_{p \in T_\omega^i} \sum_{q \in T_\omega^j} \int e^{\left(-\frac{1}{2\sigma^2}(t-p)^2\right)} e^{\left(-\frac{1}{2\sigma^2}(t-q)^2\right)} dt. \tag{3.6}$$

Verkürzt kann die Faltung der beiden Gauss-Funktionen über die Vorkommen eines K-mers an den Positionen p und q in den Sequenzen als Funktion I der Distanz $d = |p - q|$ angegeben werden. Zusammengefasst entspricht das Integral des inneren Produkts dann

$$I(d) = \sqrt{\pi}\sigma \times e^{\left(-\frac{1}{4\sigma^2}(d)^2\right)}. \tag{3.7}$$

Das innere Produkt ist wiederum eine Gauss-Funktion, deren Varianz die Summe der Varianzen der eingehenden Gauss-Funktionen ist. Da die Varianzen der eingehenden Funktionen gleich σ^2 sind, beträgt die Varianz der resultierenden Gauss-Funktion $2\sigma^2$.
Durch Einsetzen in (3.6) erhält man

$$k(\mathbf{x}_i,\ \mathbf{x}_j) = \sqrt{\pi}\sigma \sum_{\omega \in \mathcal{A}^K} \sum_{p \in T_\omega^i} \sum_{q \in T_\omega^j} e^{\left(-\frac{1}{4\sigma^2}(p-q)^2\right)}. \tag{3.8}$$

In dieser Form wird die Bedeutung des Parameters σ für den Grad der Positionsabhängigkeit klar. Für $\sigma \mapsto 0$, gehen nur die Oligomer-Vorkommen an den gleichen Positionen in beiden Sequenzen in die Summe ein, während für $\sigma \mapsto \infty$ alle Vorkommen eines Oligomers in beiden Sequenzen ohne Positionsinformation aufsummiert werden.

Um Abbildungen von Sequenzen verschiedener Länge vergleichbar zu machen, muss eine Normalisierung der Kerne vorgenommen werden. Die Normalisierung erfolgt mit Hilfe der L_2-Norm:

$$\tilde{k}(\mathbf{x}_i,\ \mathbf{x}_j) = \frac{k(\mathbf{x}_i,\ \mathbf{x}_j)}{\sqrt{k(\mathbf{x}_i,\ \mathbf{x}_i)k(\mathbf{x}_j,\ \mathbf{x}_j)}}. \tag{3.9}$$

Man erhält die aus n Trainingssequenzen resultierende normierte Kernmatrix

$$\widetilde{\mathbf{K}} = (\tilde{k}(\mathbf{x}_i,\ \mathbf{x}_j))_{i,j \in \{1,2,\ldots,n\}}. \tag{3.10}$$

Diese enthält für jede Trainingssequenz die inneren Produkte mit allen anderen Trainingssequenzen. Die Gewichtung der einzelnen Einträge für die Klassifikation wird beim Training bestimmt.

Im Folgenden werden ausschließlich die normierten Kernfunktionen verwendet, jedoch ohne in der Notation zwischen k und $\tilde{k}$ zu unterscheiden.

3.2.4 Trainieren eines Oligo-Kern-Klassifikators

Das Training erfolgt mit einer Menge von n Eingabesequenzen, deren Label bekannt ist: $\mathcal{T} = \{(\mathbf{x}_1,\ y_1),\ (\mathbf{x}_3,\ y_2),\ \ldots,\ (\mathbf{x}_n,\ y_n)\}$. Ein Label y_i kann den Wert $+1$ für die positive Klasse (*wahre* Translationsstarts) bzw. -1 für die negative Klasse (*falsche* Translationsstarts) annehmen.

Da das Training mit sehr kleinen Datenmengen möglich sein soll, ist es notwendig geeignete Regularisierungsmechanismen einzusetzen, um eine Überanpassung (*overfitting*) zu verhindern. Bekannte regularisierte Lernschemata sind sogenannte *soft-margin* SVMs [41] und die Methode der regularisierten kleinsten Quadrate (RLSQ – *regularized least square*) [31]. Beide Methode erreichen auf betrachteten Benchmark-Datensätzen eine vergleichbare Performanz [32]. In der vorliegenden Arbeit wurde die Methode der regularisierten kleinsten Quadrate verwendet (siehe dazu Abschnitt 2.5.1, S. 20), da diese für kleine Datenmengen, wie sie in dieser Arbeit verwendet werden, deutlich schneller zu berechnen ist.

Bei der RLSQ wird die Realisierung des Klassifikators durch die Minimierung des Vorhersagefehlers E unter Einbeziehung eines Regularisierungsparameters λ, mit $\lambda > 0$ durchgeführt:

$$\begin{aligned} E(\boldsymbol{\alpha}) &= \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2 + \lambda\|f\|^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}(y_i - \langle \mathbf{w},\ \phi(\mathbf{x}_i)\rangle)^2 + \lambda\|\mathbf{w}\|^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}(y_i - \phi(\mathbf{x}_i)\cdot\sum_{j=1}^{n}\alpha_j\phi(\mathbf{x}_j))^2 + \lambda\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j\langle\phi(\mathbf{x}_i),\ \phi(\mathbf{x}_j)\rangle \\ &= \frac{1}{n}\sum_{i=1}^{n}(y_i - \sum_{j=1}^{n}\alpha_j k(\mathbf{x}_i,\ \mathbf{x}_j))^2 + \lambda\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j k(\mathbf{x}_i,\mathbf{x}_j) \end{aligned} \tag{3.11}$$

Durch Einsetzen der Vektoren $\mathbf{y} = [y_1,\ y_2,\ \ldots,\ y_n]^\top$, $\boldsymbol{\alpha} = [\alpha_1,\ \alpha_2,\ \ldots,\ \alpha_n]^\top$ und der Kernmatrix $\mathbf{K} = (k(\mathbf{x}_i,\ \mathbf{x}_j))_{i,j\in\{1,\ldots,n\}}$ kann der Ausdruck entsprechend (2.12) vereinfacht werden:

$$E(\boldsymbol{\alpha}) = \frac{1}{n}(\mathbf{y} - \mathbf{K}\boldsymbol{\alpha})^\top(\mathbf{y} - \mathbf{K}\boldsymbol{\alpha}) + \lambda\boldsymbol{\alpha}^\top\mathbf{K}\boldsymbol{\alpha}. \tag{3.12}$$

Daraus ergibt sich für die Bestimmung der optimalen Lösung der Gewichtung $\boldsymbol{\alpha}$:

$$\begin{aligned} &(\mathbf{K} + \lambda n\mathbf{I})\boldsymbol{\alpha} = \mathbf{y} \\ \Longrightarrow\quad &\boldsymbol{\alpha} = (\mathbf{K} + \lambda n\mathbf{I})^{-1}\mathbf{y}, \end{aligned} \tag{3.13}$$

wobei mit $\mathbf{I} \in \mathbb{R}^{n\times n}$ die Identitätsmatrix bezeichnet ist.

Der Parameter λ beschränkt dabei die Norm der Diskriminante im Merkmalsraum, d. h. die Gewichte der für die Klassifikation relevanten Merkmale können nicht beliebig erhöht werden. Dadurch wird eine Überanpassung auf Merkmale, die nur einzelne oder wenige Trainingsdaten charakterisieren, nicht aber generelle Eigenschaften der Verteilung spiegeln, verhindert.

3.2.5 Komplexität des Oligo-Kern-Algorithmus

Für zwei Sequenzen der Länge L_1 und L_2 kann die Laufzeit zur Berechnung der Oligo-Kerne mit $O(L_1 L_2)$ angegeben werden. In den meisten Fällen liegt die reale Laufzeit jedoch deutlich unter diesem Wert, da (3.8) nur für Oligomere berechnet werden muss, die in beiden Sequenzen vorkommen. Die Berechnung wird umso aufwendiger, je mehr Oligomere in beiden Sequenzen an vielen Positionen vorkommen, was vor allem für kurze Oligomere gilt. Je länger die betrachteten Oligomere werden, desto unwahrscheinlicher wird es, dass sie in beiden Sequenzen häufig auftreten. Die Laufzeit sinkt daher rapide mit steigendem K. Hinsichtlich der Visualisierung ist K dennoch schon im Fall von DNA-Sequenzen, einem Alphabet mit vier Elementen, sehr begrenzt, da die Dimensionalität der Oligo-Funktions-Vektoren $L \times 4^K$ ist. Bei einer Länge von 200 BP und $K = 6$ sind dies über $800\,000$ Dimensionen.

Das Lernen einer geeigneten Gewichtung mit n Trainingssequenzen hat eine Komplexität von $O(n^3)$, d. h. sie steigt drastisch mit der Anzahl der Trainingssequenzen. Das Training mit bis zu 4000 Beispielen, ist in der Praxis jedoch unproblematisch. Neuere Methoden zur Steigerung der Effizienz können auch mit größeren Trainingsmengen (bis zu einigen $10\,000$ Beispielen) umgehen [42, 43]. Die Komplexität des Trainings bedeutet in viele Fällen keine Limitierung für biologische Problemstellungen, da die Menge »sicherer« Daten meist deutlich kleiner ist.

Bei der Klassifikation einer Eingabesequenz werden so viele Rechenschritte benötigt, wie Beispiele beim Training verwendet wurden, da jede Eingabesequenz mit der Kernmatrix (3.8) multipliziert wird. Für m Eingabesequenzen beträgt die Laufzeit also $O(nm)$.

3.2.6 Kombinierte Oligo-Kerne

Die Vorhersage mit Oligo-Kernen wie beschrieben basiert jeweils nur auf der Repräsentation der Sequenzen durch Oligomere einer einzelnen Länge K. Die Informationen der einzelnen Oligo-Kerne werden also separat betrachtet. Es ist anzunehmen, dass sich die Informationen ergänzen, so dass eine Kombination der Kerne die Vorhersageperformanz erhöht.

Um die Informationen verschiedener Oligo-Kerne bei einer Vorhersage gleichzeitig einzubeziehen, kann eine additive Kombination der Kerne angewendet werden. Der *kombinierte Oligo-Kern* wird analog zu (3.9) definiert als

$$\tilde{k}_{K-\text{kombi}} = \frac{1}{K} \sum_{m=1}^{K} \tilde{k}_m(\mathbf{x}_i, \mathbf{x}_j). \tag{3.14}$$

Zur Unterscheidung des kombinierten Oligo-Kerns von den Kernen, die nur eine Oligomer-Länge einbeziehen, werden letztere im Folgenden auch als »Einzel«-Kerne bezeichnet.

3.3 Anwendung des Oligo-Kern-Algorithmus auf Translationsstarts bei *E. coli* K-12

3.3.1 Datensätze

Performanz und Anwendbarkeit des Oligo-Kern-Algorithmus wurden am Beispiel prokaryotischer Translationsstarts untersucht. Obwohl es bereits eine große Anzahl vollständig sequenzierter und annotierter prokaryotischer Genome gibt, ist nur ein kleiner Teil dieser Daten experimentell belegt oder manuell überarbeitet (siehe Abschnitt 1.2.1, S. 10). Einer der wenigen zuverlässigen Datensätze in diesem Feld ist der EcoGene-Datensatz [20] des Organismus *Escherichia coli* K-12. Aus diesem Datensatz (Stand November 2003) wurde eine Menge von 722 durch Sequenzierung der N-Termini verifizierter Translationsstarts extrahiert. Um jeden annotierten Translationsstart (*wahrer* Start) aus dieser Menge wurde ein Sequenzfenster von 200 BP ausgeschnitten: 100 BP upstream (stromaufwärts, in 5′-Richtung) und 99 BP downstream (stromabwärts, in 3′-Richtung) bezogen auf die Startposition, so dass der Translationsstart an Position 101 des Ausschnittes liegt.

Als Negativ-Beispiele wurden in einem Fenster von ±60 BP um jeden der 722 verifizierten Translationsstarts Codons aus der Menge $\{ATG,\ GTG,\ TTG\}$ betrachtet (siehe Abbildung 3.2). Ein Negativ-Beispiel muss im gleichen Leserahmen liegen wie der zugehörige annotierte Translationsstart und es darf kein Stoppcodon aus der Menge $\{TAA,\ TAG,\ TGA\}$ im gleichen Leserahmen zwischen *falschem* Start und annotiertem Stopp auftreten (*in-frame* Stoppcodon). Um jeden falschen Start wurde analog zu den wahren Starts ein 200 BP Fenster der Sequenz ausgeschnitten. Nach diesem Verfahren konnte zu der Menge von 722 Positiv-Beispielen eine Menge von 854 Negativ-Beispielen (davon 576 upstream und 278 downstream eines annotierten Starts) zusammengestellt werden.

Die Begrenzung der Auswahl falscher Starts auf ein Fenster von 60 BP basiert auf einer Voruntersuchung der Vorhersagen mit dem Metaprogramm YACOP [14]. Dazu wur-

den die Vorhersagen der einzelnen in YACOP integrierten Programme mit der Annotation verglichen. Dabei zeigte sich, dass etwa zwei Drittel der inkonsistent vorhergesagten Translationsstarts für *E. coli* K-12 nicht weiter als 50 BP vom annotierten Start entfernt liegen. Diese sind offenbar die Startcodons, die am schwierigsten vom wahren Start zu unterscheiden sind, so dass mit ihnen die höchste Trennschärfe erreicht werden kann.

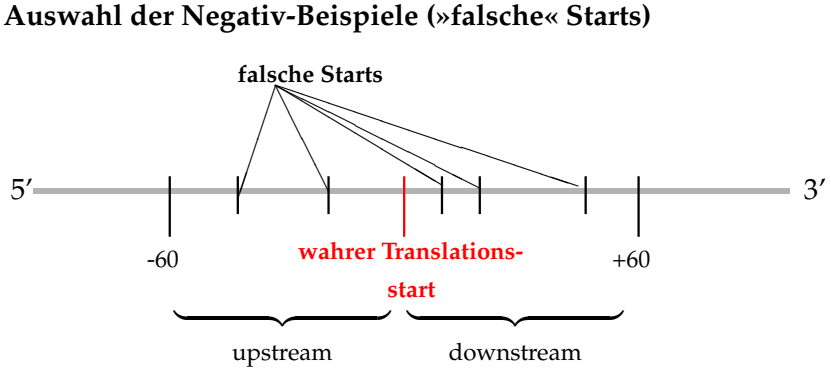

Abbildung 3.2: **Auswahl »falscher« Translationsstarts als Negativ-Beispiele. Die Negativ-Beispiele wurden in einem Fenster ±60 BP um einen annotierten Start gesucht. Jedes Negativ-Beispiel muss im gleichen Leserahmen wie der zugehörige annotierte Start liegen. Es dürfen außerdem keine *in-frame* Stoppcodons zwischen dem annotierten Genstopp und einem falschen Start auftreten.**

3.3.2 Performanz des Oligo-Kern-Algorithmus

Voraussetzung für die Anwendbarkeit des Oligo-Kern-Algorithmus zur Detektion relevanter biologischer Signale ist eine hohe Performanz bei der Vorhersage. Diese wurde auf den in Abschnitt 3.3.1 beschriebenen experimentell belegten Daten gezeigt. Der Oligo-Kern-Klassifikator wurde für Oligomere (im Folgenden gleichbedeutend mit »Oligonukleotide«) der Länge $K \in \{1, 2, \ldots, 6\}$ trainiert und auf einer nicht zum Training verwendeten Menge getestet. Für jedes K wurden 50 Läufe durchgeführt.

Der Datensatz wurde in jedem Lauf zufällig in drei Untermengen aufgeteilt: Trainingsmenge (631 Sequenzen), Validierungsmenge (378 Sequenzen) und Testmenge (567 Sequenzen). Mit der Trainingsmenge wurde zunächst ein Klassifikator trainiert, mit diesem wurden auf der Validierungsmenge die Hyperparameter σ (Varianz) und λ (Regularisierung) optimal eingestellt, so dass der Klassifikationsfehler minimiert wurde. Für die Glättung wurden die Werte $\sigma \in \{0{,}5,\ 0{,}75,\ 1,\ 1{,}5,\ 2\}$ abgetastet, für die Regularisierung die Werte $\lambda \in \{0{,}1 \cdot 0{,}9^i | i = 0,\ 1,\ \ldots,\ 100\}$. Mit den optimalen Werten für die Hyperparameter und der Vereinigung von Trainings- und Validierungsmenge wurde wiederum ein Klassifikator trainiert. Auf der bis zu diesem Zeitpunkt nicht einbezogenen Testmen-

ge wurde dann der Klassifikationsfehler bestimmt. Die Aufteilung des Datensatzes für Training, Validierung und Test ist als Übersicht in Tabelle 3.1 dargestellt.

Tabelle 3.1: **In der folgenden Tabelle ist die Aufteilung des EcoGene-Datensatzes bei Training, Validierung (Schritt 1-3) und Test (Schritt 4) des Oligo-Kern-Algorithmus zusammengefasst.**

	Datensatz		
Schritt	**Trainingsmenge**	**Validierungsmenge**	**Testmenge**
1	Trainieren eines Klassifikators	—	—
2	—	Optimieren der Hyperparameter mit dem Klassifikator	—
3	Trainieren eines neuen Klassifikators mit den optimalen Hyperparametern		—
4	—	—	Bestimmen des Klassifikationsfehlers

Der mittlere Vorhersagefehler und der Mittelwert der optimalen Varianz über alle 50 Durchläufe ist in Tabelle 3.2 zusammengefasst. Der niedrigste Fehler bei der Vorhersage liegt bei 8,9% für die Oligomerlänge $K = 3$. Für $K < 3$ und $K > 3$ steigt der Fehler monoton an, es macht daher offensichtlich keinen Sinn Kerne mit $K > 6$ zu testen.

Tabelle 3.2: **Performanz des Oligo-Kern-Algorithmus für Oligomere der Längen $K \in \{1, \ldots, 6\}$. Angegeben ist jeweils der prozentuale mittlere Fehler mit Standardabweichung auf einer unabhängigen (nicht bei Training oder Validierung verwendeten) Testmenge. Alle angegeben Werte sind über 50 Durchläufen gemittelt.**

Oligomerlänge (K)	1	2	3	4	5	6
Mittlerer (Median) Fehler	11,8 (11,8)	9,7 (9,6)	8,9 (8,7)	9,6 (9,5)	12,7 (12,6)	15,0 (15,0)
Standardabweichung	1,3	1,3	1,4	1,2	1,3	1,2
Optimales σ (mittel)	0,8	0,8	1,25	1,34	1,24	1,27

Nach den Prinzipien des Maschinellen Lernens ist anzunehmen, dass die Performanz des Klassifikators steigt, wenn eine größere Trainingsmenge repräsentativer Daten zur Verfügung steht. Daher wurden aus der kompletten GenBank-Annotation (siehe Abschnitt 1.2.1, S. 10) des Organismus *E. coli* K-12 (U00096) alle nicht-hypothetischen Proteine, d. h. alle Proteine, denen einen Funktion zugeordnet ist , für eine weitere Untersuchung extra-

hiert. Dieser Datensatz umfasst 2980 positive Beispiele. Dazu wurde analog zur Beschreibung in Abschnitt 3.3.1 eine Menge von 3968 negativen Beispielen generiert. Training, Validierung und Test erfolgten auf diesen Daten wie oben beschrieben. Dabei wurden die Ergebnisse wegen der höheren Anzahl der Trainingsdaten und der damit steigenden Laufzeit jeweils nur für 20 Durchläufe ermittelt. Die gemittelten Ergebnisse der Untersuchung sind in Tabelle 3.3 zusammengefasst.

Im Gegensatz zur Erwartung verschlechtern sich die Klassifikationsraten auf dem größeren Datensatz signifikant, was die Vermutung nahe legt, dass ein Großteil dieses Datensatzes fehlerhaft annotiert ist. Der Vergleich der experimentell belegten Annotationen des EcoGene-Datensatzes mit den entsprechenden GenBank-Einträgen zeigt, dass 10,2% dieser 722 Translationsstarts im GenBank-Datensatz falsch annotiert sind. Es ist anzunehmen, dass die Rate falsch annotierter Genstarts für den gesamten GenBank-Datensatz noch höher liegt, da hier neben falsch annotierten Starts auch ganze ORFs falsch annotiert sein können. Daher wurden für weitere Untersuchungen nur die experimentell belegten Daten verwendet.

Tabelle 3.3: **Mittlere Vorhersagefehler (Angabe in Prozent) des Oligo-Klassifikators auf dem GenBank-Datensatz für die Oligomerlängen $K \in \{1, 2, \ldots, 6\}$. Gemittelt wurde jeweils über 20 Durchläufe. Die Fehlerraten verschlechtern sich im Vergleich zu dem verifizierten EcoGene-Datensatz signifikant. Sie liegen auf dem etwa viermal größeren GenBank-Datensatz um 3,9-6,4 Prozentpunkte höher (vgl. Tabelle 3.2).**

Oligomerlänge (K)	1	2	3	4	5	6
Mittlerer Fehler	17,3	15,6	15,3	16,0	17,0	18,9

3.3.3 Vergleich der Performanz mit anderen Methoden

Die Performanz des Oligo-Kern-Algorithmus bei der Vorhersage wurde mit Wahrscheinlichkeitsmodellen, die auf relativen Oligomer-Häufigkeiten basieren, verglichen. Es wurden hierbei Modelle der Ordnung 0 (Monomere) und 1 (Dimere) einbezogen. Für entsprechende Wahrscheinlichkeitsmodelle höherer Ordnung brechen die Vorhersageraten mit einem Fehler $\approx$30% ein.

Die Vorkommen von Mononukleotiden (Wahrscheinlichkeitsmodelle der Ordnung 0) sind statistisch unabhängig. Sie entsprechen daher einfachen Markow-Modellen der Ordnung 0. Für die Modelle der Dimere trifft dies nicht zu, da hier nur die positionsspezifischen Oligomer-Vorkommen betrachtet wurden. Vergleiche der Performanz haben jedoch gezeigt, dass die Performanz der Modelle mit bedingten Wahrscheinlichkeiten auf

DNA-Sequenzen im Wesentlichen vergleichbar ist mit der der entsprechenden Markow-Modelle [44].

Bei dem Vergleich wurde außerdem der kombinierte Oligo-Kern einbezogen, bei dem die Kerne für Oligomere der Längen $K \in \{1, 2, \ldots, 6\}$ aufsummiert werden (siehe Abschnitt 3.2.6). Um die Relevanz der Positionsinformation zu zeigen, wurden die Klassifikationsraten mit verschiedenen Spektrum-Kernen ($\sigma \mapsto \infty$, $K \in \{1, 2, \ldots, 6\}$) verglichen. Hier erreicht der Kern mit $K = 2$ (Dimere) die maximale Performanz. Nur dieser wurde beim weiteren Vergleich einbezogen.

Die niedrigsten Fehlerraten werden von dem kombinierten Oligo-Kern erreicht, da dieser die Informationen aller Einzel-Kerne einbezieht. Die Raten des besten Einzel-Kernes ($K = 3$) sind jedoch nur unwesentlich schlechter. Der mittlere Klassifikationsfehler liegt hier um $0{,}8$ Prozentpunkte höher. Beide Oligo-Kerne erreichen deutlich bessere Raten als die Wahrscheinlichkeitsmodelle, deren Fehler um $3{,}3$ bzw. $3{,}4$ Prozentpunkte höher ist als für den Oligo-Kern mit $K = 3$. Werden nur Monomer-Vorkommen einbezogen, so erreichen Wahrscheinlichkeitsmodell und Monomer-Kerne (siehe Tabelle 3.2) eine vergleichbare Performanz. Bei der Klassifikation basierend auf Dinukleotid-Vorkommen sind Oligo-Kerne den einfachen Wahrscheinlichkeitsmodellen der Ordnung 1 überlegen. Der beste Spektrum-Kern liegt mit einer Fehlerrate von $44{,}6\%$ nur knapp unter dem Wert einer zufälligen Klassifikation.

Tabelle 3.4: **Vergleich der prozentualen Fehlerraten bei der Vorhersage der Oligo-Kerne mit anderen Methoden. In der Tabelle sind die Fehlerraten des Oligo-Kerns mit $K = 3$ (OK_3) und des kombinierten Oligo-Kerns (OK_{1-6}) im Vergleich mit Wahrscheinlichkeitsmodellen der Ordnung 0 (IWM_0) und 1 (IWM_1) und einem Spektrum-Kern (SP_2) für Dimere gezeigt. Die Fehlerraten sind jeweils über 50 Durchläufen mit zufällig zusammengestellten Datensätzen gemittelt.**

Methode	OK_3	$\mathrm{OK}_{1\ldots,6}$	IWM_1	IWM_2	SP_2
Mittlerer Fehler (Median)	8,9 (8,7)	8,1 (7,8)	11,4 (11,4)	11,3 (11,4)	44,6 (44,9)

3.3.4 Signaldetektion bei prokaryotischen Translationsstarts

Die Methode zur Visualisierung der diskriminativen Merkmale, wie in Abschnitt 3.2.2 beschrieben, wurde auf folgende Arten umgesetzt: Zum einen wurden Übersichten der diskriminativen Merkmale für verschiedene Oligomere der Länge K erstellt, zum anderen wurden einzelne Gewichtsfunktionen über allen betrachteten Positionen dargestellt. Die Gewichtsfunktionen der Oligomere können mit Hilfe der L_2-Norm entsprechend ihrer Relevanz für die Klassifikation geordnet werden, was im Weiteren auch als »Rangfolge«

bezeichnet wird. So ist es ohne weiteres möglich, interessante Merkmale zu identifizieren. Die entsprechenden Gewichtsfunktionen können dann im Detail über den gesamten betrachteten Bereich analysiert werden. Es ist zu beachten, dass die L_2-Norm das Auftreten eines Merkmals über alle betrachteten Positionen bewertet. Denjenigen Merkmalen, deren Funktion in der Summe die größten Schwankungen in der Amplitude aufweist, wird somit die höchste Relevanz für die Klassifikation zugeordnet.

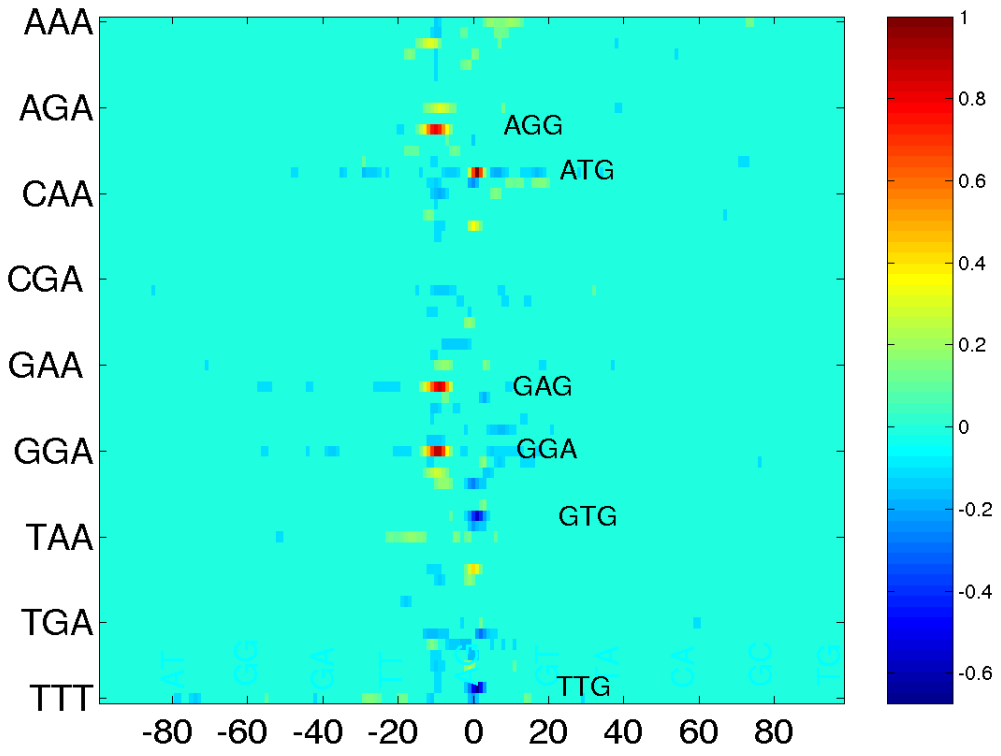

Abbildung 3.3: **Farbkodierte Darstellung der Gewichtungen aller Trinukleotide für die Klassifikation von Translationsstarts des Organismus *E. coli* K-12. Die Trinukleotide sind in alphabetischer Reihenfolge (AAA, AAC, AAG, ...TTT) angegeben. Die Positionsangaben beziehen sich auf den Translationsstart, dieser wird mit Position 0 bezeichnet. Die Positionen im negativen Bereich bezeichnen die upstream-Region bezogen auf den Translationsstart, die Positionen im positiven Bereich die downstream-Region.**

Bei der Übersichtsdarstellung (Abbildung 3.3) wird die Gewichtsmatrix $\mathbf{W}$ (3.4) als Ganzes angezeigt. Ein Zeilenvektor $\mathbf{w}_i$ stellt die Gewichte eines Kmers über alle betrachteten Sequenzpositionen dar, ein Spaltenvektor $\mathbf{w}_j$ die Gewichte aller Kmere an einer bestimmten Position. Die Relevanz eines Merkmals für die Klassifikation wird dabei mit einer Farbkodierung dargestellt. Eine Farbskala gibt den normierten Wert der Gewich-

tung zu der entsprechenden Farbe an. Geht ein Merkmal an einer Position stark in die Klassifikation zur positiven Klasse ein, so ist dies in der Grafik dunkelrot dargestellt, geht ein Merkmal an einer Position stark in die Klassifikation zur negativen Klasse ein, so ist es dunkelblau dargestellt. In Abbildung 3.3 (S. 38) sind die Gewichte für Trinukleotide ($K = 3$) bei *E. coli* K-12 in einem 200 BP Fenster um Translationsstarts gezeigt. Die Werte sind gemittelt über die optimalen Gewichtsmatrizen aus 50 Durchläufen und so normiert, dass der maximale Wert (ATG an Position 0) 1 beträgt. Außerdem wurden alle Positionen, deren absoluter Wert kleiner 0,1 ist, auf 0 gesetzt um das Rauschen zu verringern. Position 0 bezeichnet den Translationsstart, der Bereich mit negativen Positionen bezeichnet den nicht-kodierenden upstream-Bereich bezogen auf den Translationsstart, die Positionen im positiven Bereich bezeichnen den kodierenden downstream-Bereich.

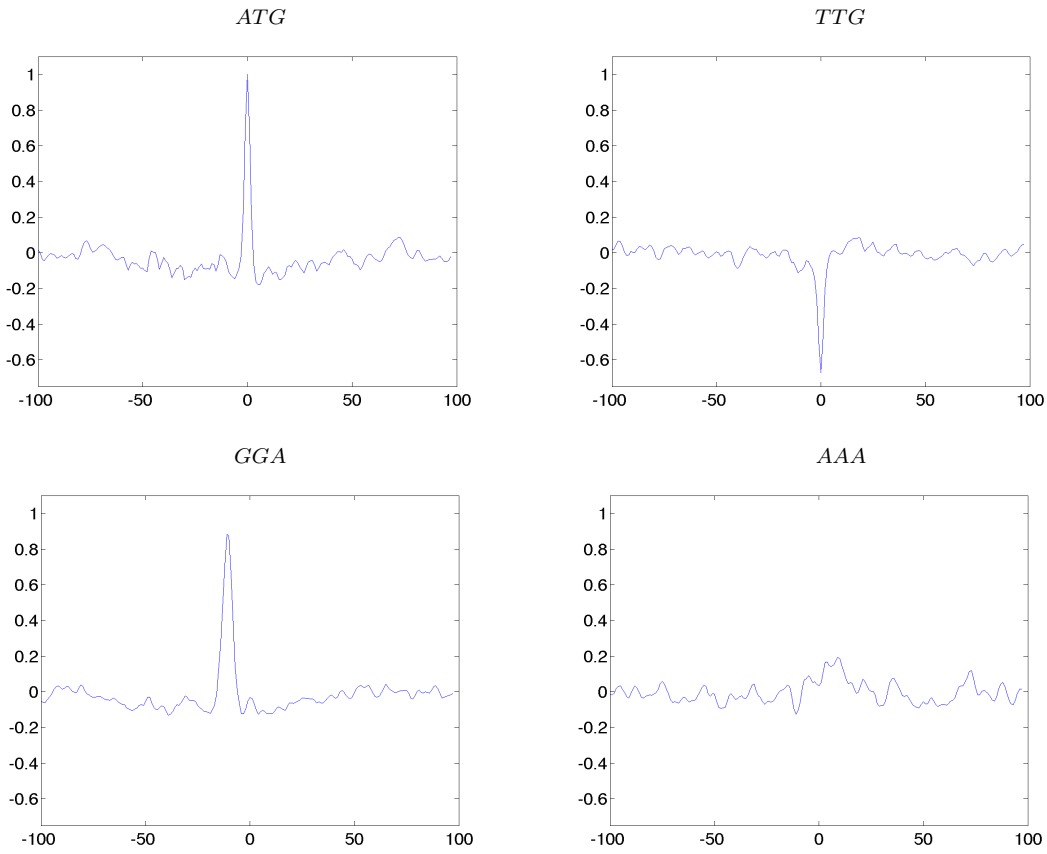

Abbildung 3.4: **Exemplarische Darstellung der Gewichtsfunktionen der Trinukleotide ATG, TTG, GGA und AAA für den Organismus *E. coli* K-12 über alle Sequenzpositionen des betrachteten 200 BP Ausschnittes. Der Translationsstarts liegt an Position 0. Upstream-Positionen sind mit negativem Vorzeichen angegeben, downstream-Positionen mit positivem. Die Werte sind über 50 Durchläufen gemittelt und skaliert auf das Maximum bei $K = 3$.**

Abbildung 3.4 (S. 39) zeigt exemplarisch die Gewichtsfunktionen der Trinukleotide ATG, TTG, GGA und AAA als zweidimensionale Funktionsplots (2D-Plots). Die Rangfolge der Oligonukleotide $K \in \{3, 4, 5, 6\}$ nach ihrer Relevanz für die Klassifikation sind als Balkendiagramm 3.5 (S. 41) gezeigt. Übersichtsdarstellungen für $K \in \{1, 2\}$, sowie weitere Darstellungen einzelner Gewichtsfunktionen als 2D-Plots und Peak-Positionen/ -Werte für alle Di- und Trimerfunktionen sind in Anhang A (S.79f) gegeben. Die Abbildungen der Gewichtsfunktionen als Übersicht, 2D-Plots, sowie Balkendiagramme und Ranglisten für alle untersuchten Oligonukleotide ($K \in \{1, 2, \ldots, 6\}$) für *E. coli* K-12 sind außerdem unter http://gobics.de/oligo_functions verfügbar.

Übersicht der diskriminativsten Oligomere

Am stärksten diskriminativ sind bei den Mononukleotiden die Gewichte von G an Position -10, gefolgt von A an Position 0. Beide gehen als positive Merkmale in die Klassifikation ein. Es folgen C mit einem Minimum an Position 0 und T ebenfalls mit einem Minimum, an Position -10. Die Übersichtsdarstellung für Mononukleotide ist im Anhang (Abbildung A.1, S. 80) gegeben.

Bei allen höheren Oligomeren dominieren offensichtlich zwei Arten von Signalen: Zum einen Signale die im Zusammenhang mit der Shine-Dalgarno-Sequenz (SD) [45] stehen (GG, GGA, $AGGA$ usw.) und ein lokales Maximum in der Region -12...-7 aufweisen sowie Signale, die im Zusammenhang mit dem Startcodon stehen. Oligomere wie AT, ATG, $ATGGC$ usw. weisen ein lokales Maximum an Position 0 auf, dies kann auf die Überrepräsentation von ATG als Startcodon (mehr als 90%) zurückgeführt werden. Häufigste Codons nach dem Translationsstart sind offenbar AAA und GCT was aus den hohen positiven Gewichten der Hexanukleotide $ATGAAA$ und $ATGGCT$, sowie der Pentanukleotide $ATGGC$ und $ATGAA$ zu schließen ist. Die Balkendiagramme (3.5, S. 41) zeigen die große Relevanz dieser Oligonukleotide für die Klassifikation. Die positiven Gewichte der Trinukleotide TAT und CAT an Position -1 sind auch mit der Überrepräsentation von ATG als Startcodon zu erklären. Sie deuten aber auch an, dass es eine Präferenz zu T und C an Position -1 gibt, was ebenso bei höheren Oligomeren zu erkennen ist. So ist $CATGA$ auf Rang sieben der diskriminativsten Pentanukleotide und $TTATG$ auf Rang neun. Oligomere, welche TTG oder GTG enthalten, werden entsprechend negativ gewichtet, wenn die Trinukleotide an der Position des Translationsstarts stehen, da TTG und GTG bei den wahren Starts unterrepräsentiert sind.

Neben den starken Signalen, fallen bei den höheren Oligomeren drei weitere Signaltypen auf: Poly-A in der downstream-Region direkt nach dem Translationsstart, Poly-T

in der Region -25 bis -15 BP und eine AT-Box um Position -18. Die beiden ersten Signale sind insbesondere bei den Trinukleotiden zu erkennen, aber auch bei den Tetranukleotiden, mit $AAAA$ und $TTTT$ auf Rang neun bzw. zehn und Hexanukleotiden mit $TTTTTT$ auf Rang neun. Die AT-Box ist ebenfalls am deutlichsten bei den Trimeren erkennbar, wo AAT, ATA und TAA an Position -18 ein lokales Maximum haben.

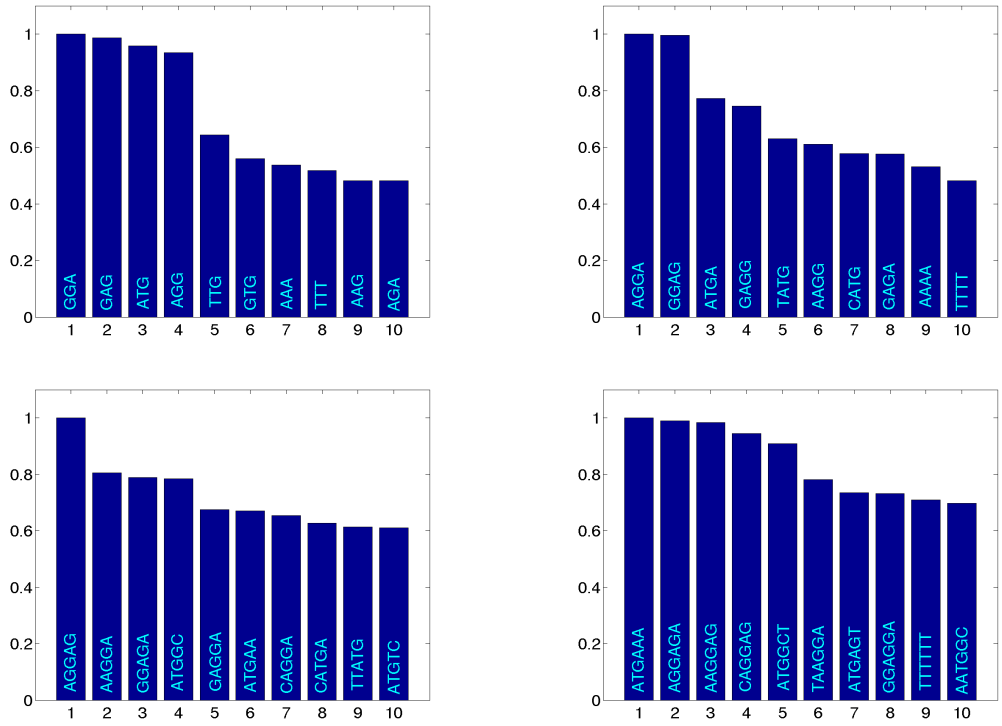

Abbildung 3.5: **Balkendiagramm der Rangordnung für Trinukleotide, Tetranukleotide, Pentanukleotide und Hexanukleotide bezüglich ihrer Relevanz für die Klassifikation. Die Rangordnung basiert auf der L_2-Norm (3.5) der Gewichtsfunktionen in der Region um Translationsstarts des Organismus *E. coli* K-12. Die Höhe der Balken entspricht der Norm. Die Werte sind gemittelt über 50 Durchläufen und skaliert auf den maximalen Wert für die jeweilige Oligomerlänge.**

Diskussion der detektierten Signale

Wie bereits im vorigen Abschnitt beschrieben stehen die diskriminativsten Signale bezüglich der Klassifikation von Translationsstarts, wie erwartet mit der SD-Sequenz und dem Startcodon in Zusammenhang, den prominentesten Signalen der Translationsinitiation. Da ATG als Startcodon in den positiven Beispielen stark überrepräsentiert ist (90,1% der Translationsstarts im EcoGene-Datensatz), werden die beiden anderen Startcodons GTG und TTG (7,4% und 1,9% im EcoGene-Datensatz) stark negativ gewichtet.

Die SD-Sequenz, von Shine und Dalgarno für *E. coli* als vier oder mehr aufeinander folgende Basen des Regulären Ausdrucks $RGGRGGTGAT$ ($R = A$ oder G) in der Region -15...-6 BP upstream des Translationsstarts beschrieben [45], wird von dem Oligo-Klassifikator klar detektiert. Auch die Variabilität des Motivs und der Position stimmen mit publizierten Merkmalen der RBS in *E. coli* überein [46, 47]. Das zeigt sich unter anderem in der Breite der Spitze der Gewichte von GGA. Sie ist deutlich breiter als beispielsweise die Spitze des Startcodons ATG, welches keine Variation in der Position aufweist (vgl. Abbildung 3.4, S. 39). In der Rangfolge relevanter Merkmale (Balkendiagramm) nehmen GGA und GAG vor ATG die Ränge eins und zwei ein, obwohl der ATG-Peak an Position 0 eine etwas höhere Amplitude aufweist. Das ist darauf zurückzuführen, dass die Summe der Amplituden über alle betrachteten Positionen bei GGA und GAG höher ist als bei ATG.

Neben diesen Merkmalen sind auch schwächere Signale zu erkennen: Ein lokales Maximum weist auf das Vorkommen von A direkt nach dem Startcodon auf (Positionen 3...18). Diese downstream-Poly-A-Box ist bereits als Element zur Modulation des Expressionslevels bekannt [48, 49, 50]. Sie ist insbesondere bei den Trinukleotiden zu erkennen, aber auch bei höheren Oligomeren zeigt sich eine deutliche Präferenz zu A direkt nach dem Translationsstart, beispielsweise bei den Hexanukleotiden mit $ATGAAA$ als diskriminativstem Merkmal des Oligo-Klassifikators $K = 6$. Ein weiteres lokales Maximum weist auf die Bedeutung von Poly-T in der upstream-Region ≈-20 BP hin. Dieses könnte im Zusammenhang mit der Umgebung des Promotors stehen, die als AT-reiche Region bekannt ist (siehe auch Abschnitt 1.1.3, S. 6). Als AT-reich wird auch die 3-4 BP umfassende Region direkt upstream der RBS beschrieben, diese steht mit dem S1-Protein in Zusammenhang, welches bei *E. coli* eine wichtige Funktion bei der Erkennung der mRNA durch das Ribosom spielt [51, 52]. Das Signal ist bei den Trimeren AAT, ATA und TAA als lokales Maximum an Position -18 erkennbar.

Bei der Analyse einzelner Oligomer-Funktionen lassen sich noch subtilere Signale erkennen, wie beispielsweise der leichter Abfall der Gewichtsfunktionen bei den Trimeren ATG und GGA in der Region bis ±40 BP um den Translationsstart. Dies ist auf die Auswahl der negativen Beispiele zurückzuführen. Da die Negativ-Beispiele in den angrenzenden Regionen (±60 BP) von wahren Translationsstarts ausgewählt wurden, enthalten sie, entsprechend verschoben, die Signale der Translationsinitiation, die hier aber als negative Gewichte in die Klassifikation eingehen.

Die vom Oligo-Kern-Algorithmus detektierte Signale spiegeln sich in der Abbildung eines Sequenz-Logos der unmittelbaren Umgebung der experimentell verifizierten Translationsstarts von *E. coli* (Abbildung 3.6). Das Sequenz-Logo basiert auf positionsabhängi-

gen Mononukleotid-Häufigkeiten, wobei die Größe eines Symbols jeweils der Häufigkeit des Auftretens an der Position entspricht. Die starken Signale des Startcodons (ATG an Position 0) und RBS sind deutlich als zusammengesetztes Signal erkennbar, ebenfalls zu erkennen sind die Poly-A-Downstreambox und die AT-Box vor der RBS. Sehr schwache Signale, Signale mit großer Positionsunsicherheit und negativ gewichtete Signale werden im Logo-Plot nicht sichtbar. Es muss auch die Einschränkung gemacht werden, dass mit Logo-Plots nur grobe Annahmen über längerer Signale erlauben, da nur Mononukleotid-Häufigkeiten dargestellt sind.

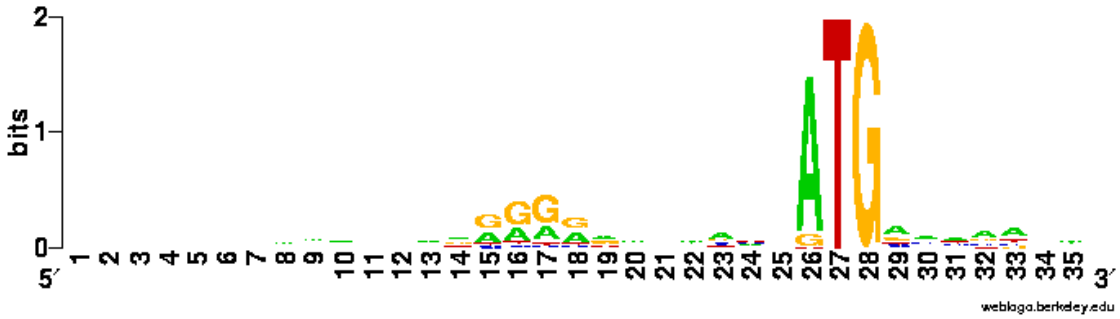

Abbildung 3.6: **»Logo-Plot« [53] des Sequenzbereichs -25…+10 (bezogen auf den Translationsstart) experimentell verifizierter TIS des EcoGene-Datensatzes, erstellt mit dem Programm WEBLOGO [54]. Position 1 in der Abbildung entspricht Position -25 bezogen auf den Translationsstart. Der Translationsstart befindet sich an Position 26.**

3.3.5 Schlussfolgerungen aus der Analyse von Translationsstarts mit dem Oligo-Kern-Klassifikator

Die Analyse der Gewichte zeigt, dass die relevanten Merkmale für die Translationsinitiation bei *E. coli* K-12 hauptsächlich in der Region -30…+10 BP bezogen auf den wahren Translationsstart liegen. Bei der Vorhersage von Translationsstarts ist es daher offenbar ausreichend, diesen Ausschnitt um TIS-Kandidaten zu erfassen. Dieses Wissen und die Kenntnis relevanter Muster bei der Translationsinitiation kann bei der Entwicklung und Optimierung von Vorhersageprogrammen, sowie der Evaluierung automatischer Vorhersagen einbezogen werden.

Der Vergleich mit der Performanz anderen Methoden (siehe Abschnitt 3.3.2, S. 34 und 3.3.3, S. 36) zeigt die Relevanz der Positionsinformation für die Klassifikation von Translationsstarts. Die positionsunabhängigen Spektrum-Kerne erreichen mit Abstand die schlechtesten Klassifikationsraten. Die vollständig positionsabhängigen Wahrscheinlichkeits-modelle der Ordnung 0 zeigen eine vergleichbare Performanz wie der Monomer-

Kern (Tabelle 3.4, S. 37). Oligo-Kerne höherer Ordnung sind einfachen Wahrscheinlichkeitsmodellen offenbar überlegen. Die Performanz der Dimer-Kerne im Fallbeispiel ist besser als die der Wahrscheinlichkeitsmodelle 1-ter Ordnung. Dabei benötigt der Oligo-Kern-Klassifikator weniger gesicherte Beispiele für das Training, so dass auch die Analyse längerer Signale möglich ist. Die Glättung der Positionsinformation erlaubt es, mit dem Oligo-Kern-Algorithmus auch Signale mit Positionsunsicherheit zu erfassen.

Der Monomer-Kern scheint im Vergleich zu Wahrscheinlichkeitsmodellen 0-ter Ordnung für das Problem der Identifikation von Translationsstarts überdimensioniert. Hinzu kommt, dass die meisten bekannten biologischen Signale nicht oder nur schwer auf Monomer-Basis zu charakterisieren sind. Daher ist der Monomer-Kern weniger geeignet für derartige Analysen. Basierend auf Oligomeren höherer Ordnung lassen sich biologische Signale mit der ihnen eigenen Variabilität jedoch sehr gut modellieren und analysieren.

Die Möglichkeit, die Länge betrachteter Oligomere und den Grad ihrer Positionsabhängigkeit einzustellen, macht den Oligo-Kern-Algorithmus zu einem flexiblen Werkzeug. Er lässt sich gleichermaßen auf die Detektion anderer Signale, beispielsweise von eukaryotischen Spleißstellen (*splice sites*) [55, 56] und Transkriptionsfaktorbindestellen (nicht veröffentlichte Studien), anwenden. Die Anwendung ist dabei nicht auf die Analyse von DNA-Sequenzen beschränkt, auch die Detektion von Signalen in Proteinsequenzen ist denkbar.

3.4 Weiterentwicklung der Oligo-Kerne

Durch die Auswahl eines Kerns wird die Metrik zwischen den Eingaben festgelegt, was sich entscheidend auf die Performanz des Klassifikators auswirkt. Igel *et al.* (2007) entwickelten ein effizientes Verfahren zur Modellselektion (*modell selection*), dessen Performanz am Beispiel der Oligo-Kerne demonstriert wird [57]. Dabei steht die Optimierung der Glättung im Vordergrund, die als Teil der Modellselektion behandelt wird. Als Vergleichsmaß für die Kerne wird ein sogenanntes *Kernel-Target-Alignment* [57, 58] berechnet.

Bei der Untersuchung wird der kombinierte Oligo-Kern mit $K = 1, 2, \ldots, 6$ verwendet, der eine bessere Performanz zeigt als die »Einzel«-Kerne (siehe Abschnitte 3.2.6 , S. 32 und 3.3.3, S. 36). Dabei wird für jeden Einzel-Kern ein individueller Glättungsparameter σ_i verwendet, so dass für jede Oligomerlänge K die Positionsabhängigkeit individuelle eingestellt werden kann. Beispielsweise kann im kombinierten Kern für Hexamere ($K = 6$) nur die Häufigkeit, ohne Positionsinformation gezählt werden ($\sigma \mapsto \infty$), während Trimerauftreten nur als ähnlich bewertet werden, wenn sie an ähnlichen Positionen

in beiden Sequenzen auftreten. Die Optimierung der Glättungsparameter wird hier nicht für jeden Kern isoliert berechnet, wie bisher beim kombinierten Oligo-Kern, sondern mit einem Gradienten-Aufstieg der Kernel-Target-Alignment-Funktion gleichzeitig für alle betrachteten K.

Mit diesem Ansatz konnte gezeigt werden, dass die optimale Glättung für das Beispiel prokaryotischer Translationsstart von der Länge der betrachteten Kmere abhängt. Die Performanz des kombinierten Oligo-Kerns wird mit Markow-Modellen mit Pseudocounts und dem Locality Improved Kernel [59] verglichen. Dabei zeigt der Oligo-Kern eine bessere Performanz.

Eine mögliche Weiterenwicklung der Oligo-Kerne sind Motiv-Oligo-Kerne bzw. eine additive Kombination des kombinierten Oligo-Kern mit Motiv-Oligo-Kernen, wobei der Begriff Motiv kurze Sequenzmuster bezeichnet. Während Oligo-Kerne wie bisher beschrieben alle Oligomere der Länge K einbeziehen, werden Motiv-Kerne über eine begrenzte Menge von Mustern $A_{\text{Motiv}} \subset \mathcal{A}^*$ definiert. Durch die Auswahl der Motive kann zusätzlich biologisches Vorwissen über das Problem eingebracht werden. Wie für die Motiv-Kerne ist es denkbar die Oligo-Kerne mit weiteren Kernen zu kombinieren.

Kapitel 4

Ein unüberwachtes Verfahren zur Vorhersage von Translationsstarts

Zu den meisten Fragestellungen in der Biologie gibt es wenige experimentell gesicherte Daten. Im Fall der Genannotation bei Prokaryoten ist dies nicht anders. Viele der Annotationen, die in öffentlichen Datenbanken zugänglich sind, beruhen allein auf automatischen Vorhersagen und sind stark fehlerbehaftet. Das gilt insbesondere für die Annotation der Translationsstarts (siehe Abschnitt 1.2, S. 8). So ist es vor allem schwierig für neusequenzierte Genome, verlässliche Klassifikatoren mit überwachten Verfahren zu trainieren. Es war daher ein Hauptziel dieser Arbeit, Möglichkeiten zur (Re-)Annotation von Translationsstarts mittels unüberwachter Verfahren zu untersuchen und eine effiziente, in der Praxis anwendbare Software zu entwickeln, die eine zufriedenstellende Vorhersage ermöglicht.

In jüngster Vergangenheit wurden einige Ansätze entwickelt, um die Vorhersage von Translationsstarts (TIS – *translation initiation site*) zu verbessern. Während ältere Ansätze Kenntnis von den Mustern (*Motiven*) der Ribosombindestellen (RBS) einbeziehen, die eine Analyse der speziesspezifischen 16S-rRNA erfordern [60, 61, 11], werden die meisten Parameter neuerer Ansätze mit unüberwachten Verfahren angepasst [12, 13, 62], so dass bei der Anwendung im Prinzip kein Vorwissen zu den Charakteristika des Eingabegenoms nötig sind. Diese Ansätzen verwenden jedoch nach wie vor einige sequenzspezifische Parameter, die nicht mit unüberwachten Verfahren gelernt werden. Darunter fällt beispielsweise die erwartete Länge der RBS, ihr Abstand zum Startcodon, die Anzahl in Betracht gezogener RBS-Motive (SD-Sequenzen) oder Annahmen über die Häufigkeiten der Startcodons. Solche Parameter werden mit empirischen Werten voreingestellt, die auf Studien an gut untersuchten Organismen wie *Escherichia coli* oder *Bacillus subtilis* beruhen, die

beide einen moderaten GC-Gehalt (Anteil an Guanin und Cytosin im Genom) von etwa 50% haben. Die Charakteristika dieser Genome können aber keineswegs als generelle Charakteristika aller prokaryotischen Genome betrachtet werden. Es wurde gezeigt, dass der Informationsgehalt der RBS stark von der systematischen Gruppe abhängt [63]. Viele Archaeen weisen keine SD-Sequenz auf, sondern verwenden Signale, die mehr den eukaryotischen Initiationsfaktoren gleichen [64]. Hinzu kommt, dass über den bei weitem größten Teil der Prokaryoten (mind. 90%) bislang noch gar nichts bekannt ist, da viele Organismen mit derzeitigen Methoden im Labor nicht kultivierbar sind. Es besteht also das Risiko, durch die Voreinstellung von Parametern, die im Zusammenhang mit speziellen Charakteristika der Sequenz stehen, suboptimale Ergebnisse zu erzielen.

Im Gegensatz zu bisherigen Methoden werden bei dem im Folgenden vorgestellten Verfahren zur Vorhersage von Translationsstarts, keine sequenzspezifischen Annahmen einbezogen. Trotz der Generalität des Ansatzes sind die Ergebnisse vergleichbar gut, in vielen Fällen sogar besser als die vorheriger Ansätze. Insbesondere auf Genomen mit hohem GC-Gehalt (von über 60%) zeigt der Ansatz eine gute Performanz.

Das im Folgenden vorgestellte Verfahren zur Nachbearbeitung (*Postprocessing*) der Annotation von Translationsstarts [2] ist in dem Programmpaket TICO (Translation Initiation site COrrection) [3, 4] implementiert und öffentlich zugänglich unter
http://tico.gobics.de.

4.1 Klassifikation von TIS-Kandidaten mit einem unüberwachten Lernverfahren

In diesem Teil der Arbeit wird die Klassifikation potentieller Translationsstarts (TIS-Kandidaten) durch ein iteratives Clusterverfahren untersucht. Als Eingabe wird eine Vorhersage kodierender Regionen mit einer initialen Annotation potentieller Translationsstarts vorgegeben, wie sie beispielsweise mit dem Genvorhersageprogramm GLIMMER [10] erstellt werden kann. Diesen TIS-Kandidaten wird zunächst das Label *strong* (stark) zugeordnet. In einem Suchfenster (siehe Abbildung 4.1, S. 49) um die vorhergesagten Translationsstarts werden weitere TIS-Kandidaten gesucht, welchen das initiale Label *weak* (schwach) zugeordnet wird. Für alle Kandidaten gilt, sie müssen im gleichen Leserahmen liegen wie das zugehörige Stoppcodon und es dürfen keine in-frame-Stoppcodons zwischen TIS-Kandidat und annotiertem Stopp auftreten. Mit Hilfe des Clusterverfahrens wird für die Kandidaten jeweils ein quantitativer »Gütewert« (*Score*) berechnet. Der Score

dient dazu in jeder Iteration die TIS-Kandidaten zu klassifizieren und entsprechend der Klassifikation die Label *weak* und *strong* neu zu verteilen.

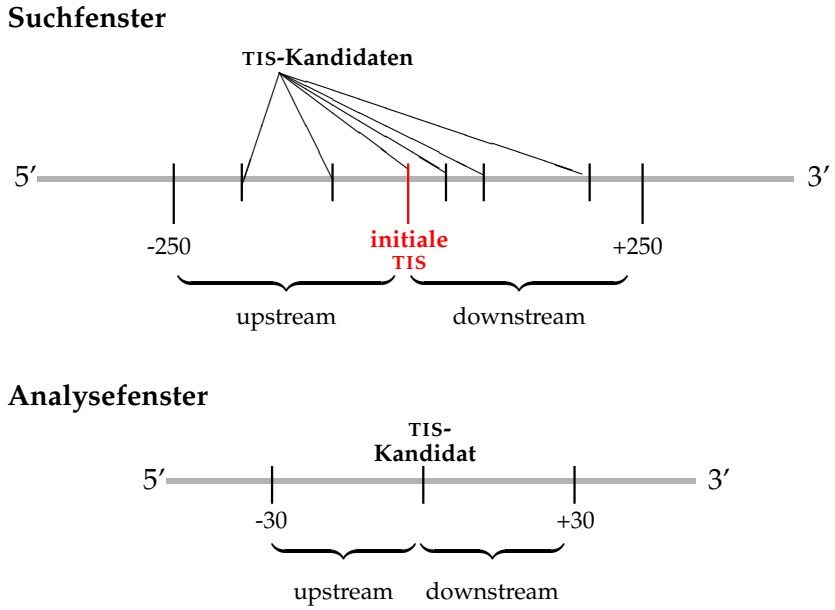

Abbildung 4.1: **Such- und Analysefenster des Programms TICO mit Standardwerten. In der oberen Abbildung ist die Auswahl der TIS-Kandidaten gezeigt. In einem Suchfenster um die »initiale« TIS werden Startcodons lokalisiert, welche im gleichen Leserahmen wie der annotierte Stopp liegen, ohne dass ein in-frame-Stoppcodon zwischen TIS-Kandidaten und annotiertem Stopp auftritt. In der unteren Abbildung ist die Extraktion eines Sequenzfensters (*Analysefenster*) um einen TIS-Kandidaten gezeigt.**

4.2 Schema der unüberwachten Klassifikation

Die TIS-Kandidaten gehören zu einer von zwei Klassen, der Klasse der starken Kandidaten (Label *strong*) oder der Klasse der schwachen Kandidaten (Label *weak*). Beide Klassen von TIS-Kandidaten werden jeweils durch ein Wahrscheinlichkeitsmodell 2-ter Ordnung ($K = 3$) repräsentiert. Die Wahrscheinlichkeiten basieren auf den positionsabhängigen Trinukleotid-Häufigkeiten in Sequenzausschnitten der Länge L um die TIS-Kandidaten. Der Grad der Positionsabhängigkeit ist dabei flexibel, wie schon für den Oligo-Kern-Algorithmus beschrieben wurde. Zum einen lassen sich so die Positionsunsicherheiten von Signalen modelliert, zum anderen Null-Wahrscheinlichkeiten verhindern. In der Praxis wird die Glättung mit Hilfe einer stochastischen Glättungsmatrix berechnet (siehe Abschnitt 4.2.1, S. 52). Im Folgenden sei die Anzahl aller TIS-Kandidaten mit n bezeich-

net, die Anzahl der betrachteten ORFs wird mit N angegeben, N_i ist die Anzahl der TIS-Kandidaten, die dem ORF i zugeordnet sind, so dass gilt $n = \sum_i^N N_i$. Die Label werden durch die Binärwerte $h_{ij} \in \{0, 1\}$ repräsentiert. Ist $h_{ij} = 1$, dann ist der $j$te Kandidat des $i$ten ORFs der Kategorie *strong* zugeordnet, ist $h_{ij} = 0$, so ist er der Kategorie *weak* zugeordnet. Die Klassifikation der Kandidaten unterliegt dabei der Nebenbedingung, dass maximal ein Kandidat eines ORFs der Kategorie *strong* zugeordnet sein darf:

$$\sum_{j=1}^{N_i} h_{ij} \leq 1,\ i \in \{1,\ \ldots,\ N\}. \tag{4.1}$$

Durch diese Nebenbedingung wird der biologische Sachverhalt modelliert, dass prokaryotische Gene in aller Regel nur einen wahren Translationsstart aufweisen.

Die Kodierung der Trinukleotid-Vorkommen erfolgt ebenso wie die der Label binär, so dass ein Eintrag $x_{kl}^{ij} \in \{0, 1\}$ in der Datenmatrix $\mathbf{X} \in \mathbb{R}^{L4^3 \times n}$ angibt, ob Trinukleotid k an Position l des $j$ten Kandidaten von ORF i vorkommt. Ein Eintrag p_{kl}^s der Wahrscheinlichkeitsmatrix $\mathbf{P}_s$ bezeichnet die Wahrscheinlichkeit des Auftretens eines Trinukleotides k an Position l einer Kandidatensequenz der Kategorie *strong*. Entsprechend bezeichnet ein Eintrag p_{kl}^w der Wahrscheinlichkeitsmatrix $\mathbf{P}_w$ die Wahrscheinlichkeit des Auftretens eines Trinukleotides k an Position l einer Kandidatensequenz der Kategorie *weak*. Die Wahrscheinlichkeiten werden für jede Iteration mit der aktuellen Klassifikation geschätzt. Die Kategorie eines Kandidaten wird nur geändert, wenn dadurch der Score steigt. Die Iteration wird so lange fortgeführt, bis die Klassifikation der Kandidaten sich nicht mehr ändert.

Das Ziel des Lernschemas ist die Maximierung der Funktion

$$F(\mathbf{P}_s,\ \mathbf{P}_w, \{h_{ij}\}) = \sum_{i,j} h_{ij} \sum_{k,l} x_{kl}^{ij} \log p_{kl}^s + \sum_{i,j} (1 - h_{ij}) \sum_{k,l} x_{kl}^{ij} \log p_{kl}^w. \tag{4.2}$$

Dabei werden die logarithmierten Wahrscheinlichkeiten in jedem Iterationsschritt (d. h. für gegebene h_{ij}) neu berechnet. Der Algorithmus konvergiert hinsichtlich der monoton ansteigenden Summe der Scores sofern keine Null-Wahrscheinlichkeiten auftreten. Dies wird im Allgemeinen durch die Glättung der Positionsinformation verhindert.

Verkürzt kann die Berechnung der Positionsgewichtsmatrix (PWM – *positional weights matrix*) $\mathbf{W} \in \mathbf{R}^{4^3 \times L}$ als

$$\mathbf{W} = \log \widetilde{\mathbf{P}}_s - \log \widetilde{\mathbf{P}}_w, \tag{4.3}$$

angegeben werden, wobei mit $\widetilde{\mathbf{P}}_{q \in s,w} \in \mathbb{R}^{4^3 \times L}$ die geglätteten Wahrscheinlichkeiten $\mathbf{P}$ bezeichnet sind. Die Berechnung der Glättung mittels einer Glättungsmatrix wird in Ab-

schnitt 4.2.1, S. 52 beschrieben. Der Score eines Kandidaten wird als Produkt der PWM mit dem Merkmalsvektor des Kandidaten berechnet.

Der Clusteralgorithmus lautet also wie folgt:

Initialisierung:
Annotierte TIS in der Eingabe erhalten das Label *strong*, alle weiteren Kandidaten erhalten das Label *weak*.

Iteration:

1. Schätzen der positionsspezifischen Trinukleotid-Wahrscheinlichkeiten, jeweils für starke und schwache Kandidaten
2. Berechnen der Positionsgewichtsmatrix (PWM) entsprechend der aktuellen Klassifikation.
3. Berechnen der Scores für alle Kandidaten
4. (Re-)Klassifikation der Kandidaten, Neuzuordnung der Label: Das Label *strong* erhält derjenige Kandidat eines ORFs mit dem maximalen positiven Score, alle anderen Kandidaten dieses ORFs erhalten das Label *weak*. Wenn kein Kandidat des ORFs einen positiven Score hat, so erhalten alle das Label *weak*.

Abbruchbedingung:
Der Abbruch erfolgt, wenn sich die Verteilung der Label bei der Reklassifikation nicht mehr verändert oder eine maximale Anzahl von Iterationen erreicht ist.

Obwohl der Algorithmus auch ohne weitere Beschränkung konvergiert, wurde zur Verkürzung der Laufzeit eine maximale Anzahl von 20 Iterationen festgesetzt. Bei allen Untersuchungen lagen nach 20 Iterationen die relativen Veränderungen in den Scorewerten nur noch unter einem Promille, und es gab keinen messbaren Einfluss mehr auf die Ergebnisse der TIS-Vorhersage. Alle Ergebnisse wurden in Testläufen mit 50 Iterationen bestätigt.

Dieser Clusteralgorithmus ist sehr ähnlich zu dem bekannten EM-Algorithmus [34] (siehe dazu Abschnitt 2.5.2, S. 22). Die Zuordnung von Datenpunkten zu Clustern ist im Fall des TICO-Algorithmus jedoch deterministisch, während sie bei dem allgemeinen EM-Algorithmus probabilistisch ist. Weiterhin unterscheidet beide die Nebenbedingung auf den Zuordnungsvariablen (4.1).

4.2.1 Berechnung der Glättung

Um die Konvergenz des Clusterverfahrens zu garantieren, muss verhindert werden, dass die Wahrscheinlichkeiten der Oligomer-Vorkommen zu klein werden, da dann die Werte der Scores gegen unendlich gehen. Dies erfolgt durch eine Glättung, wie sie schon für den Oligo-Kern-Algorithmus beschrieben ist (Abschnitt 3.2.1, S. 27). Realisiert wird die Glättung mit einer diskretisierten Gauss-Funktion, deren Werte durch eine Glättungsmatrix repräsentiert werden. Abbildung 4.2 (S. 52) zeigt eine graphische Darstellung der Glättungsmatrix. Ein Eintrag s_{ij} der Matrix in $\mathbf{S} \in \mathbb{R}^{L \times L}$ wird mit Hilfe einer normalisierten

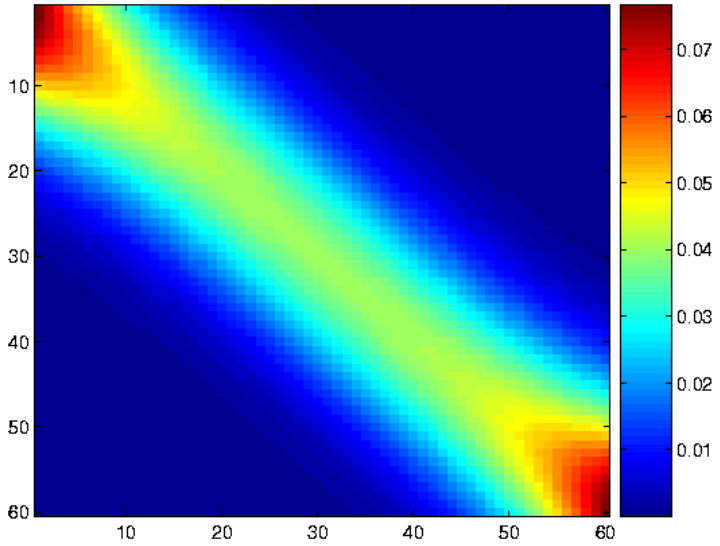

Abbildung 4.2: **Graphische Darstellung der Glättungsmatrix (4.4)**

Gauss-Funktion berechnet:

$$s_{ij} = \frac{e^{\left(-\frac{1}{2\sigma^2}(i-j)^2\right)}}{\sum_l e^{\left(-\frac{1}{2\sigma^2}(l-j)^2\right)}} \qquad i,j,l \in \{1,\ldots,L\}. \tag{4.4}$$

Die Glättung eines positionsabhängigen Wahrscheinlichkeitsmodells der Ordnung $K-1$ für eine Sequenz der Länge L kann als einfaches Matrixprodukt der Glättungsmatrix mit der Wahrscheinlichkeitsmatrix $\mathbf{P} \in \mathbb{R}^{4^K \times L}$ berechnet werden:

$$\widetilde{\mathbf{P}} = \mathbf{P} \cdot \mathbf{S}. \tag{4.5}$$

Da sowohl $\mathbf{S}$ als auch $\mathbf{P}$ stochastische Matrizen sind, ist die resultierende Matrix $\widetilde{\mathbf{P}}$ ebenfalls eine stochastische Matrix, d. h. die Einträge sind jeweils positiv und die Spaltensummen betragen 1.

Mit dem oben beschrieben Verfahren erhält man keine gleichmäßige Glättung über alle Positionen. In Abbildung 4.2 ist deutlich erkennbar, dass die Glättung an den Rändern schwächer ist, da dort weniger Nachbarpositionen einbezogen werden können. Um diese Randeffekte zu vermindern, werden bei der Berechnung die ersten drei und die letzten drei Positionen der Sequenzen nicht betrachtet.

Ein anderes Verfahren, mit dem Null-Wahrscheinlichkeiten verhindert werden können, ist die Addition von konstanten Werten (*Pseudocounts*) zu den Matrixeinträgen. Die Glättung mit Hilfe einer Gauss-Funktion wirkt jedoch vorallem lokal. Aus diesem Grund ist sie besser für die Modellierung solcher Korrelationen geeignet, die in erster Linie zwischen benachbarten Positionen bestehen. Nach biologischem Wissen ist dies bei den Signalen der prokaryotischen Translationsinitiation der Fall.

4.2.2 Automatische Adaption des Glättungsfaktors

Die Einstellung eines Glättungsfaktors durch den Benutzer setzt ein gewisses Vorwissen über die Natur der Daten und Erfahrung mit dem Verfahren voraus. Der optimale Grad der Glättung hängt vom Eingabegenom ab. Für sehr kleine Genome oder Genome, deren Signale sehr heterogen sind, ist eine stärkere Glättung der Wahrscheinlichkeiten als die mit dem vorgegebene Standardwert $\sigma = 0{,}5$ vorteilhaft. Um ein verlässliches Verfahren zu realisieren, das keines speziellen Vorwissens des Benutzers bedarf, wurde der Algorithmus erweitert, so dass neben der manuellen Vorgabe eines fixen Glättungsfaktors auch eine automatische Anpassung durch das Programm möglich ist. In einer Untersuchung der Vorhersageperformanz wurde gezeigt, dass mit der automatischen Anpassung des Glättungsfaktors keine signifikante Verschlechterung der Vorhersagen gegenüber dem optimalen Wert einhergeht (siehe dazu Abschnitt 4.5.3, S. 68). Es sei an dieser Stelle darauf hingewiesen, dass der optimale Wert im Anwendungsfall in der Regel nicht bekannt ist.

Um den Grad der Glättung automatisch einzustellen, wird eine Kreuzvalidierung angewendet, welche die Stärke der Abgrenzung der beiden Klassen für ein gegebenes Modell bestimmen soll. Die finale Abgrenzung beider Klassen in einer Vorhersage ist anhand eines Histogramms der Score-Verteilungen im Anhang gezeigt (Abschnitt B.1, S. 88). Für die Analyse wird die Nebenbedingung (4.1) ausgenutzt, dass maximal ein Kandidat eines ORFs der Kategorie *strong* angehören darf. Dieses Vorgehen beruht auf der Annahme, dass, wenn ein Kandidat der Kategorie *strong* eine wahre TIS ist, alle zum selben ORF gehörigen Kandidaten der Kategorie *weak* keine wahren TIS sein können. Es werden daher bei der Analyse nur solche Kandidaten der Kategorie *weak* einbezogen, die einem ORF

zugeordnet sind, zu dem es einen Kandidaten der Kategorie *strong* gibt.

Als Maß für die Performanz der Klassifikation während eines Iterationsschrittes wird eine ROC-Analyse (*receiver operating characteristics*) mit AUC-Kriterium (*area under curve*) verwendet. Eine nähere Beschreibung der ROC-Analyse ist im folgenden Abschnitt (S. 55f) gegeben. Die ROC-Analyse wird dabei für jedes Modell und jeden Glättungswert $\sigma \in \{0{,}25, 0{,}3, 0{,}35, \ldots, 1{,}0\}$ durchgeführt. Das AUC-Kriterium als Maß für die Güte der Klassifikation für einen gegebenen Wert σ bezieht sowohl die Raten der Falsch-Positiven als auch die der Falsch-Negativen ein. Sie ist daher ein geeignetes Maß, die Performanz der Klassifikation bei einer sehr unausgeglichenen Datenlage zu bestimmen, bei der, wie im Fall der TIS-Klassifikation, die Anzahl der Negativen deutlich höher ist als die der Positiven.

Die AUC wird in einer zehnfachen Kreuzvalidierung geschätzt: Die TIS-Kandidaten werden so in zehn zufällige Gruppen aufgeteilt, dass jede Gruppedas gleiche Verhältnis von Kandidaten der Kategorien *strong* und *weak* enthält. Neun der Gruppen werden zum jeweiligen Schätzen der PWM verwendet, während eine Gruppe ausschließlich zur Bewertung des Modells mit Hilfe der berechneten PWM-Scores dient. Dieser Vorgang wird zehnmal wiederholt, wobei jede der zehn Gruppen einmal zur Berechnung der Scores verwendet wird. Der mittlere AUC-Testwert für alle zehn Gruppen wird dann zur Bewertung des aktuellen Glättungsfaktors verwendet.

Zusammenfassung des Algorithmus mit Adaption des Glättungsfaktors:

Initialisierung:
$\sigma = 0{,}5$

Iteration:

1. Schätzen der Wahrscheinlichkeiten und Klassifikation für den aktuellen Wert von σ wie zuvor beschrieben (S. 51)
2. 10fache-Kreuzvalidierung mit $\sigma \in \{0{,}25, 0{,}3, 0{,}35, \ldots, 1{,}0\}$ für die im ersten Schritt berechnete Klassifikation
3. Setzen von σ auf den Wert, für den die AUC (gemittelt über 10 Durchläufe) maximal ist

Abbruchbedingung:
Der Abbruch erfolgt, wenn sich der Wert σ in Schritt 3 nicht mehr ändert.

Ein alternativer Ansatz für die automatische Einstellung der Glättung wäre die Vorgabe eines Glättungspriors, der dann mit Hilfe eines speziellen Lernschemas angepasst wird.

Die Einstellung eines Glättungspriors, wie in [65] beschrieben, ist jedoch deutlich komplizierter und rechenaufwendiger.

Receiver-Operating-Characteristics-Analyse mit Area-Under-Curve-Kriterium

Die ROC-Analyse (*receiver operating characteristics*) ist ein Verfahren mit dessen Hilfe sich die Performanz verschiedener Vorhersagemethoden beschreiben und vergleichen lässt. Dabei werden die Parameter TPR (*true positive rate* – Rate der Wahr-Positiven) und FPR (*false positive rate* – Rate der Falsch-Positiven) gegeneinander aufgetragen. Jeder Punkt der resultierenden Kurve entspricht dabei einem bestimmten Schwellenwert des Verfahrens, für welchen die Raten bestimmt werden.

TPR und FPR sind definiert als

$$\text{TPR} = \frac{\#\text{TP}}{\#\text{TP} + \#\text{FN}} \tag{4.6}$$

$$\text{FPR} = \frac{\#\text{FP}}{\#\text{TN} + \#\text{FP}}, \tag{4.7}$$

mit #TP (*true positives*) ≡ Anzahl der Wahr-Positiven, #FP (*false positives*) ≡ Anzahl der Falsch-Positiven, #FN (*false negatives*) ≡ Anzahl der Falsch-Negativen, #TN (*true negatives*) ≡ Anzahl der Wahr-Negativen. Tabelle 4.1 zeigt eine Zusammenfassung der Parameter für den speziellen Fall der TIS-Vorhersage.

Tabelle 4.1: **Die Bedeutung der Parameter bei der Berechnung von TPR und FPR am Beispiel der TIS-Vorhersage**

	Referenz	
Vorhersage	wahre TIS	falsche TIS
strong TIS	TP	FP
weak TIS	FN	TN

Die Entscheidung, ob ein Kandidat durch das Verfahren als wahr vorhergesagt wird, hängt in diesem Fall von dem Gütewert (hier dem PWM-Score) ab, der dem Kandidaten zugeordnet ist. Liegt der Score über dem gesetzten Schwellenwert, so wird der Kandidat als wahr (hier *strong*) vorhergesagt, andernfalls als falsch (hier *weak*). Die Bestimmung der Werte TP, TN, FP und FN basiert auf dem Vergleich der Vorhersage mit einer Referenz. Im Fall der automatischen Adaption des Glättungsfaktors ist die Referenz die bestehende Klassifikation in dem jeweiligen Iterationsschritt.

Für die Vergleichbarkeit verschiedener Verfahren, in diesem Fall verschiedener Glättungsfaktoren, wird das AUC-Kriterium (*area under curve*-Kriterium) einbezogen. Dieses wird als Integral der Auftragung TPR vs. FPR für alle betrachteten Schwellenwerte berechnet. Je größer der resultierende Wert ist, desto besser ist das Verfahren, in diesem Fall der Klassifikator.

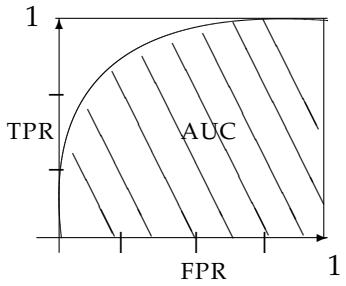

Abbildung 4.3: **Skizziertes Beispiel einer ROC-Kurve. Die Fläche unter der Kurve (AUC – *area under curve*) wird als Bewertungskriterium verwendet. Je größer die Fläche ist, desto besser ist das Verfahren.**

4.3 Implementierung

Das Verfahren wurde in dem Programm TICO [3, 4] implementiert und getestet. TICO ist über eine Webschnittstelle unter http://tico.gobics.de/ verfügbar. Auf der angegebenen Seite wird TICO auch als eigenständiges (*stand-alone*) Kommandozeilenprogramm bereitgestellt, so dass es in automatische Annotationssysteme integriert werden kann. Die Benutzerschnittstelle basiert auf JAVA, der Cluster-Algorithmus ist unter MATLAB® implementiert und als MATLAB®-Kompiler-generiertes Programm integriert. Das Kommandozeilenprogramm steht für verschiedene Betriebssysteme und Architekturen bereit.

4.3.1 TICO-Webserver

Der TICO-Webserver ist in einem Apache TOMCAT-Container [66] realisiert, einer Umgebung zum Ausführen von JAVA-Kode. Die Implementierung der Webschnittstelle basiert auf JAVA Servlets [67] und JAVA Server Pages (JSP) [68] Technologie. Auf der Eingabeseite (*submission page*) kann der Benutzer eine Genomsequenz und eine initiale Annotation hochladen. Derzeit werden als Eingabeformate für Sequenzen das FASTA-Format und für die Annotation die Formate GLIMMER2.x und SimpleCoord unterstützt (siehe Abschnitt 4.3.3, S. 58). In der Download-Sektion werden außerdem einige Perl-Skripte zur Konvertierung weiterer Annotations-Formate in ein von TICO lesbares Eingabeformat zur Verfügung gestellt.

Der Benutzer hat über die Schnittstelle die Möglichkeit, einige Parameter für die Vorhersage anzupassen, beispielsweise die Größe der Suchfenster für TIS-Kandidaten um die initiale Annotation (siehe Abschnitt 4.3.2). Andernfalls werden bewährte Standardwerte voreingestellt. Die Ergebnisse der von TICO bestimmten Vorhersage werden per E-mail an den Benutzer übermittelt. In der derzeitigen Version ist die Ausgabe in GFF (*General Feature Format*) [69], SimpleCoord und einem GLIMMER-ähnlichen Format möglich (siehe Abschnitt 4.3.3, S. 60).

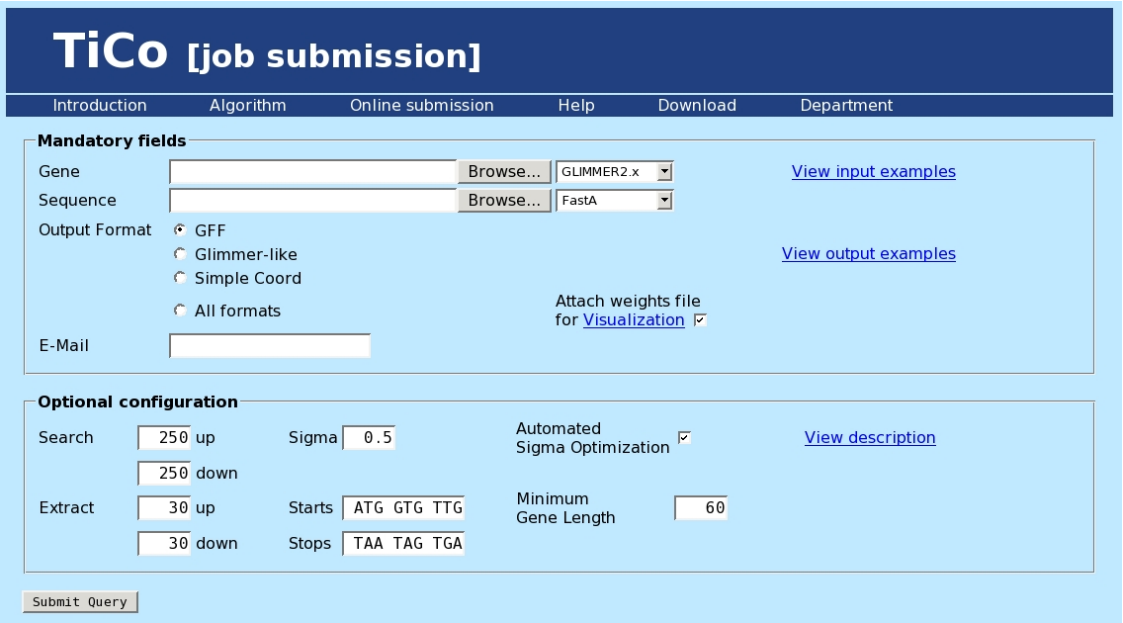

Abbildung 4.4: **Screenshot (»Bildschirmfoto«) der Eingabeseite (*submission page*) des TICO-Webservers. Die Sequenzdatei und eine initiale Annotation müssen hier durch den Benutzer hochgeladen werden. Es besteht die Möglichkeit, verschiedene Ausgabeformate zu wählen und einige Parameter anzupassen (*optional configuration*). Um die Ergebnisse zu erhalten, muss der Benutzer außerdem eine E-mail-Adresse angeben.**

4.3.2 Parameter

Der Benutzer hat sowohl über die Webschnittstelle, als auch über das Kommandozeilenprogramm die Möglichkeit, TICO durch Setzen von Parametern zu konfigurieren. Wichtig sind hier insbesondere die Einstellungen zum Suchen (*search*) von TIS-Kandidaten und zum Extrahieren (*extract*) der Analysefenster um die Kandidaten.

Biologischen Erkenntnissen zufolge befinden sich relevante Signale der Translationsinitiation im Bereich unmittelbar up- und downstream des Translationsstarts. Dies zeigte sich auch bei der Evaluation von Translationsstarts mit Hilfe des Oligo-Kern-Algorithmus (Abschnitt 3.3.5, S. 43). Die Standardwerte für des Analysefensters bei TICO wurde daher

mit ± 30 BP voreingestellt. Maximal kann die Region sowohl up- als auch downstream auf 100 BP erweitert werden. Das Suchfenster, welches um die initialen TIS-Kandidaten nach weiteren Kandidaten durchsucht wird, ist mit einem Wert von ± 250 BP voreingestellt und kann bis auf ± 500 BP erweitert werden. Dem Programm wird außerdem eine minimale Genlänge von 60 BP vorgegeben. TIS-Kandidaten, welche einen geringeren Abstand zum zugehörigen Stoppcodon aufweisen werden nicht berücksichtigt. Die minimale Genlänge kann auf jeden ganzzahligen Wert ≥ 0 eingestellt werden. Weiterhin hat der Benutzer die Möglichkeit die automatische Adaption des Glättungsparameters abzustellen und manuell einen fixen Wert $\sigma \in \{0{,}1,\ 0{,}2,\ 0{,}3,\ \ldots,\ 2{,}0\}$ vorzugeben. Eine manuelle Vorgabe des Glättungsparameters sollte allerdings nur in Ausnahmefällen, beispielsweise für sehr kleine Datensätze, notwendig sein. Im Kommandozeilenprogramm können außerdem die Mengen angegeben werden, die als Start- und Stoppcodons betrachtet werden, was für die Vorhersage in Spezies notwendig ist, die vom Regelfall abweichende Codons verwenden.

4.3.3 Ein- und Ausgabeformate

Eingabedaten

Die genomische Sequenz muss dem Programm im FASTA-Format übergeben werden, dem einfachsten Standardformat für biologische Sequenzen. Per Definition enthält eine FASTA-Datei nur die Sequenz, in Groß- oder Kleinschreibung, und eine Kopfzeile (*header*), welche mit einem > beginnt. Die Kopfzeile kann Informationen über die Sequenz enthalten, wie den Namen des Organismus, verschiedene IDs, die Länge der Sequenz usw.

Beispiel einer Sequenz im FASTA-Format

```
>gi|53717639|ref|NC_006350.1| B. pseudomallei K96243 chr. 1, complete sequence
TCATTCGCGCCACCGCAGGCACTGCTGGCGAACCGTTTCGTTCAGCGATTTTTCCTGCTTGAAGATCGTC
CGCAGCCACGAGGCGTCGTTGACGCGCGCCTTCGCGAGCGCGCCGATCTCGTCGAGCGCCGCGCGCGAGC
CGAGCGCCGCCGCGTGCGGCGCGATGCGATCGAGCGTGTCGAGGATGTCCTCGGCGATCGTCCGGCGCTC
GCCCGTCTGCGGATTCACGCAGGTGCCCTCGAGTCCGAAGCGGCATGCCTCGAAACGGTTGAACGTGTAG
ACGAGATAGTCGTCCTCCGACAGCTTGAGCGGCCGGTCGATCAGCAGATAGCGCGCGAGCGTCTGGATGT
                                  ...
```

Die initiale Annotation kann in der aktuellen Version in den Formaten GLIMMER2.x oder SimpleCoord angegeben werden. Letzteres ist ein Format, das im Rahmen dieser Arbeit eingeführt wurde und das im Folgenden erläutert wird. Im GLIMMER-Format können vor

der eigentlichen Annotation weitere Informationen über die Vorhersage durch GLIMMER angegeben sein, beispielsweise die Entwicklung der Vorhersage in den einzelnen Iterationsschritten. Diese Informationen sind für die TICO-Vorhersage irrelevant, das Einlesen beginnt daher erst ab der Zeile

»Putative Genes:«,

die als essentieller Schlüssel für den TICO-Parser in der Datei vorhanden sein muss. Alle Angaben vor dieser Zeile werden von dem Programm ignoriert.

Das GLIMMER-Format ist spaltenweise organisiert, die einzelnen Spalten sind jeweils durch Leerzeichen voneinander getrennt. Die Spalten enthalten der Reihenfolge nach: Die ID des Gens, die Position der TIS, die Position vor dem Stoppcodon, Stranginformation und Länge des Gens, optional noch eine Spalte mit zusätzlichen Informationen.

Beispiel für das GLIMMER2.x-Format

```
Putative Genes:
    1     1248        4  [-1 L=1245]  [DelayedBy #2 L=1038]
    2     2402     1164  [-3 L=1239]
    3     2458     2964  [+1 L= 507]  [LowScoreBy #4 L=179 S=2]
    4     2992     2786  [-2 L= 207]  [ShorterThan #3 L=179 S=97]
    5     3018     5108  [+3 L=2091]  [LowScoreBy #6 L=1992 S=0]
    6     5249     3117  [-3 L=2133]  [OlapWith #5 L=1992 S=99]
    8     5368     5763  [+1 L= 396]
   10     7185     5872  [-1 L=1314]  [ShadowedBy #11]  [DelayedBy #12 L=87]
   11     5872     7368  [+1 L=1497]  [Contains #10]  [LowScoreBy #12 L=212 S=6]
                             ...
```

Das SimpleCoord-Format wird vor allem wegen seiner Einfachheit verwendet. Jede Zeile beginnt mit einem >, gefolgt von einer ID, die dem Gen zugeordnet ist. Durch Unterstriche getrennt folgen die Koordinaten des Gens und seine Orientierung im Strang. Optional können nach der Orientierung – abgetrennt durch einen Unterstrich – weitere Informationen angegeben sein. Die Koordinaten des 3'-Endes beziehen sich auf das gesamte Gen (inklusive Stoppcodon), im Gegensatz zur eher unüblichen Notation von GLIMMER2.x, wo die Koordinaten der kodierenden Region (ohne Stoppcodon) angegeben sind.

Beispiel für das SimpleCoord-Format

```
>BPS0001_1_1116_-
>BPS0002_1161_2375_-
>BPS0004_3114_5168_-
>BPS0005_5488_5766_+
```

```
>BPS0006_5869_7209_-
>BPS0007_7528_8013_-
>BPS0008_8395_10668_+
        ...
```

Ausgabedaten

Die Ergebnisse der Vorhersage von TICO kann der Benutzer in folgenden Formaten ausgeben lassen: GFF (*general feature format*), SimpleCoord-Format und ein GLIMMER2.x-ähnliches Format. Die Ausgaben im GLIMMER2.x-ähnlichen Format und im SimpleCoord-Format gleichen denen der Eingabe. Es sind hier lediglich jedem Eintrag der berechnete TICO-Score und die Verschiebung der Startposition durch TICO zugefügt.

Die Ausgabe im GFF-Format entspricht den Spezifikationen des Sanger Institutes [69] und enthält zu jedem Eintrag in dieser Reihenfolge folgende Informationen: Sequenzname, Quelle der Annotation, Bezeichner des Eintrag, Startposition, Endposition, Score, Strang, Leserahmen, Attribute und optionale Kommentare (siehe Syntaxangabe im Anhang B.1, S.85). Die Spalten sind jeweils durch ein Tabulatorzeichen getrennt. Spalten ohne Eintrag werden mit einem Punkt gekennzeichnet. In der Ausgabedatei von TICO sind sowohl die ursprüngliche Annotation als auch die Positionen der reannotierten Starts enthalten, so dass ein direkter Vergleich der Ergebnisse mit der initialen Annotation möglich ist. Die Startpositionen aus der initialen Annotation sind mit dem Bezeichner `CDS` (*coding sequence* – kodierende Sequenz) gekennzeichnet, durch TICO reannotierte Einträge mit dem Bezeichner `REANNCDS`.

Zu allen Einträgen ist der von TICO berechnete Score angegeben. Bei reannotierten Startpositionen wird außerdem die Verschiebung des Starts bei der Reannotation (*shift*) angegeben. TIS-Kandidaten mit einem negativen Score sind zusätzlich mit dem Bezeichner `weak tis` versehen (siehe Beispiel zur Ausgabe im GFF-Format).

Beispiel für eine Ausgabe im GFF-Format

```
##gff-version 2
##Type DNA
NC_006350 Glimmer/tico REANNCDS  1    1116  22.892487  -  . shift 132
NC_006350 Glimmer/tico CDS  1    1248  22.892487  -  .
NC_006350 Glimmer/tico REANNCDS  1161  2375  21.028835  -  . shift 27
NC_006350 Glimmer/tico CDS  1161 2402  21.028835  -  .
NC_006350 Glimmer/tico REANNCDS  2653  2967  -6.372408  +  . shift 195;note "weak tis"
NC_006350 Glimmer/tico CDS  2458  2967  -6.372408  +  . note "weak tis"
                              ...
```

Die GFF-Ausgabe kann mit dem Tool ARTEMIS [70] visualisiert werden (siehe Abbildung 4.5). Durch Einfügen der Zeile `colour_o_REANNCDS = 1` in die ARTEMIS-Konfigurationsdatei werden alle Einträge mit dem Bezeichner `REANNCDS` grau dargestellt.

Abbildung 4.5: **Visualisierung einer TICO-Vorhersage mit ARTEMIS. Die initiale Annotation ist in hellblau dargestellt, Einträge, welche durch TICO reannotiert wurden, grau.**

4.4 Visualisierung der Gewichte

Die aktuelle Version von TICO wurde um das Tool WeightsVis [4] erweitert. Mit WeightsVis kann der Benutzer die diskriminativen Merkmale für eine Klassifikation visualisieren. Die Visualisierung basiert auf der Positionsgewichtsmatrix (PWM), welche mit dem Clusterverfahren berechnet wird. Dargestellt ist der Beitrag eines Merkmals – in diesem Fall der Trinukleotid-Vorkommen innerhalb der betrachteten Sequenzausschnitte – zur Berechnung des TICO-Scores. Ein hohes positives Gewicht zeigt an, dass ein Trinukleotid an der entsprechenden Position als starker Hinweis für eine wahre TIS eingeht, ein hohes negatives Gewicht entsprechend für eine falsche TIS.

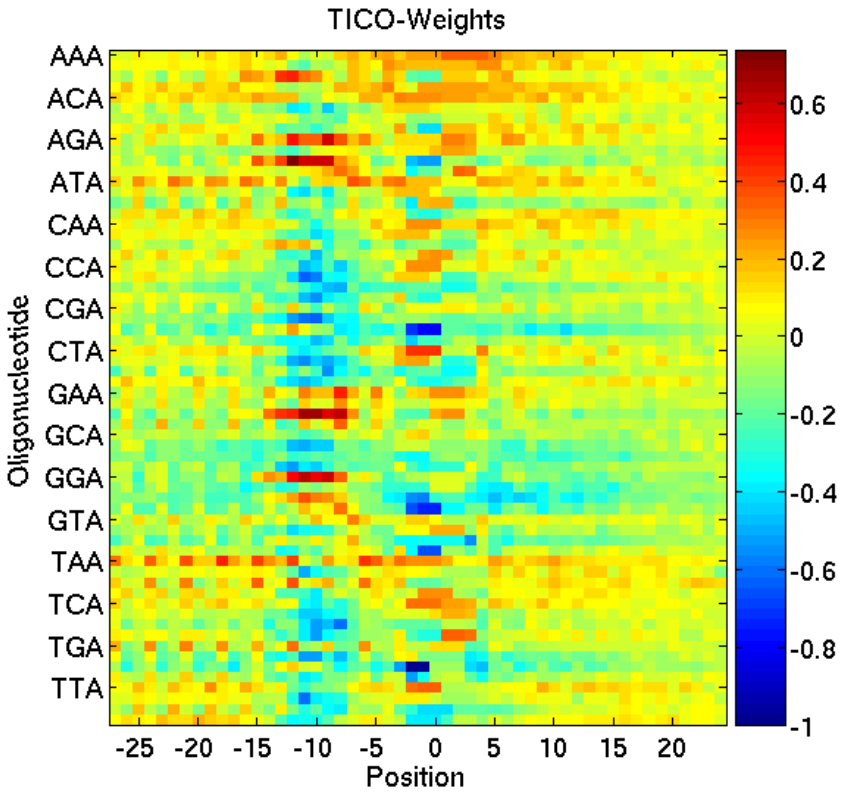

Abbildung 4.6: **Visualisierung der Gewichte bei der Klassifikation von TIS-Kandidaten des Organismus *E. coli* K-12 mit den Standardeinstellungen. Die Visualisierung wurde mit dem Tool WeightsVis aus dem TICO-Paket erstellt.**

Die Visualisierung ist ähnlich gestaltet wie die der Oligo-Funktionen: Die PWM wird in eine Farbdarstellung transformiert, so dass die Gewichte eines Trinukleotides (Zeilen) an allen Positionen eines Sequenzausschnittes (Spalten) um einen TIS-Kandidaten dargestellt sind (siehe Abb. 4.6). Positive Gewichte sind rot dargestellt, negative blau. Eine Farbskala zeigt die numerischen Werte, die mit den Farben assoziiert sind. Die numerischen Werte einzelner Positionen können außerdem durch »Mausklick« im Bild erfragt werden. Zur näheren Betrachtung steht eine Zoomfunktion zur Verfügung. Das Abspeichern des Bildes ist in 12 Formaten möglich, u. a. in EPS (*ecapsulated postscript*), PDF (*portable document format*), PNG (*portable network graphics*) und JPG (*joint photographic experts group*).

Die Visualisierung ermöglicht dem Benutzer eine Interpretation des TIS-Modells. Einerseits können biologisch relevante Signale und ihre Position in Bezug auf den Translationsstart identifiziert werden. Andererseits erlaubt die Visualisierung Rückschlüsse auf die Güte des Modells und damit auf die Qualität der Eingabedaten. Wie sich schon für

den Oligo-Kern-Algorithmus in Abschnitt 3.3.4 (S. 37) herausstellte, zeigt auch Abbildung 4.6 deutlich, dass für den Organismus *E. coli* vor allem Signale, die mit der Shine-Dalgarno-Sequenz ($AGGAG$ in der Region ≈ -12 bis -4) [45] in Zusammenhang stehen, als Hinweis für wahre TIS in die Klassifikation eingehen. Diese, durch viele Quellen belegten, Signale sind deutlich identifizier- und lokalisierbar.

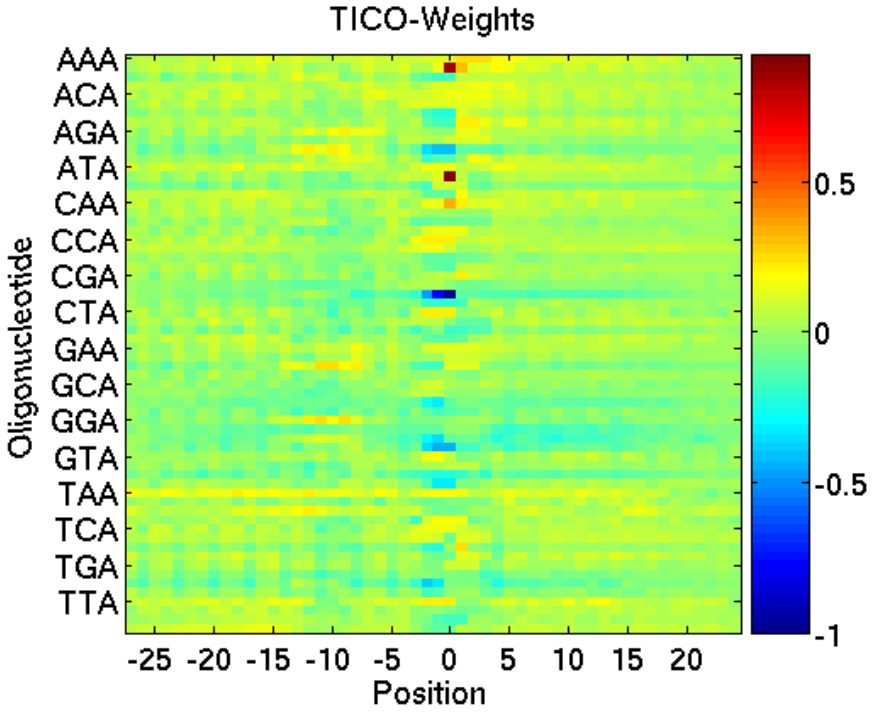

Abbildung 4.7: **Visualisierung der Gewichte bei der Klassifikation von TIS-Kandidaten des Organismus *Ralstonia solanacearum* mit den Standardeinstellungen. Die Visualisierung ist in diesem Fall deutlich weniger aussagekräftig als die der *E. coli*-Gewichte, da das Signal-Rausch-Verhältnis deutlich schlechter ist. Die Visualisierung wurde mit dem Tool WeightsVis aus dem TICO-Paket erstellt.**

In anderen Fällen ist die Visualisierung weniger aussagekräftig. In Abbildung 4.7 sind die Gewichte für den Organismus *Ralstonia solanacearum* dargestellt. Wie bei den meisten Genomen mit hohem GC-Gehalt ist das Signal-Rausch-Verhältnis von starken TIS zu schwachen TIS deutlich schlechter als für Genome mit moderatem GC-Gehalt, da Genome mit hohem GC-Gehalt weniger Stoppcodons aufweisen. Die Vorhersage für solche Genome kann durch TICO dennoch in vielen Fällen signifikant verbessert werden. Generell muss für die Visualisierung die Einschränkung gemacht werden, dass nur stärkere Signale sichtbar sind. Für Genome, deren Signale sehr heterogen sind, oder für Datensätze mit einem schlechten Signal-Rausch-Verhältnis, wie dem Beispiel von *R. solanacearum*, ist eine Interpretation der visualisierten Gewichte schwierig.

4.5 Anwendung von TICO zur Vorhersage von Translationsstarts

4.5.1 Datensätze

Die Evaluation des Tools TICO erfolgte auf Basis von Datensätzen, die standardmässig zum Vergleich der Vorhersageperformanz von Genvorhersageprogrammen für prokaryotische Genome verwendet werden: Dem bereits für die Evaluation des Oligo-Kern-Algorithmus verwendeten EcoGene-Datensatz [20] des Organismus *Escherichia coli* K-12 [21] so wie den experimentell charakterisierten Genen des Organismus *Bacillus subtilis* Stamm 168 [22, 71, 61], den sogenannten »*non-y*«-Genen. Beide Genome haben mit 50,8%, bzw. 43,5% einen moderaten GC-Gehalt. Außerdem wurde TICO hinsichtlich seiner Robustheit auf »schwierigen« Datensätzen getestet. Dies sind in der Genomannotation bei Prokaryoten vor allem Genome mit hohem GC-Gehalt, allgemeiner: Genome, die sich hinsichtlich ihrer Merkmale von den oben genannten »Standard-Genomen« unterscheiden.

Aus dem EcoGene-Datensatz (Stand Oktober 2004) wurden alle Gene extrahiert, die mit dem Schlüsselwort *verified* (bestätigt) gekennzeichnet sind. Der resultierende Datensatz umfasst 854 Gene, deren Translationsstarts biochemisch durch Sequenzierung des N-Terminus belegt sind. Aus der Annotation von *Bacillus subtilis* [22] wurden alle Gene verwendet, die mit dem Schlüsselwort *non-y* gekennzeichnet sind, so dass ein weiterer Datensatz von 1248 experimentell belegten Genen vorlag. Die Translationsstarts im *non-y*-Datensatz sind manuell überarbeitet, im Gegensatz zum EcoGene-Datensatz sind aber nicht alle Starts biochemisch belegt.

Die Performanz von TICO auf Genomen mit hohem GC-Gehalt wurde anhand der Genome von *Pseudomonas aeruginosa* PAO1 [23, 24], *Burkholderia pseudomallei* K96243 (Chromosom 1 und Chromosom 2), sowie *Ralstonia solanacearum* GMI1000 (Chromosom und Megaplasmid) gezeigt. Der GC-Gehalt dieser Organismen liegt zwischen 66 und 69%.

Für das Bakterium *Pseudomonas aeruginosa* PAO1 wurde der aktuelle Datensatz des *Pseudomonas Community Annotation Project* (PseudoCAP) [24] als Referenz verwendet. Die Mitglieder von PseudoCAP überarbeiten die im Jahre 2000 veröffentlichte Annotation manuell, mit dem Ziel, deren Qualität und Informationsgehalt zu verbessern. Der Datensatz des PseudoCAP umfasst 5647 annotierte Gene (Stand Mai 2005). Um eine möglichst verlässliche Annotation zusammenzustellen, wurden aus diesem Datensatz alle Gene entfernt, die mit den Schlüsselwörtern *hypothetical* (hypothetisch), *uncharacterized* (nicht charakterisiert) oder *unknown* (nicht bekannt) gekennzeichnet sind, so dass der verwendete Referenzdatensatz 3281 Gene mit annotierter Funktion umfasst.

Tabelle 4.2: **Zusammenfassung der Details zu den Referenzdatensätzen für die Evaluation von TICO. Angegeben ist jeweils der Name des Organismus, die GenBank-Accession-Nummer (GBK-Acc.), die Größe des Genoms in MBP, der GC-Gehalt (% GC), die Anzahl der Gene in der GenBank-Annotation (#GBK), die Anzahl der von GLIMMER vorhergesagten Gene (#GLIMMER) und die verwendete Referenzannotation (Ref.) mit der Anzahl der Einträge.**

Organismus	GBK-Acc.	Größe	% GC	#GBK	#GLIMMER	Ref. (Anzahl)
E. coli K-12	[NC_000913]	4,6 MBP	50,8	4237	5063	EcoGene (854)
B. subtilis	[NC_000964]	4,2 MBP	43,5	4106	5068	GBK *non-y* (1248)
P. aeruginosa PAO1	[NC_002516]	6,2 MBP	66,6	5647†	8620	P.CAP‡ (3281)
B. pseudomallei K96243 Chr. 1	[NC_006350]	4,1 MBP	67,7	3399	6576	GBK
B. pseudomallei K96243 Chr. 2	[NC_006351]	3,2 MBP	68,6	2329	4773	GBK
R. solanacearum Chr.	[NC_003295]	3,7 MBP	67,0	3440	5385	GBK
R. solanacearum Plasmid	[NC_003296]	2,1 MBP	66,9	1676	2797	GBK

† Gesamte PseudoCAP-Annotation

‡ Nur Gene der PseudoCAP-Annotation mit annotierter Funktion

Die Genome der Organismen *B. pseudomallei* und *R. solanacearum* wurden zum einen auf Grund ihres hohen GC-Gehaltes ausgewählt, zum anderen, weil allgemein angenommen wird, dass sie eine große Anzahl von *Repeats* (Wiederholungen), Prophagen, *Inserts* (Einfügungen) und Fremdgenen enthalten und wegen ihrer Heterogenität ein besonderes Problem für die automatisierte Genvorhersage darstellen. Die Sequenzierung und Annotation des Erregers der Melioidose (Pseudorotz), *B. pseudomallei*, erfolgte durch das Sanger Institute [69]. Die GenBank-Annotation (GBK) umfaßt 3399 Gene auf Chromosom 1 und 2329 Gene auf Chromosom 2. Sie basiert auf vergleichender Genomanalyse mit Hilfe des Artemis Comparative Tools (ACT) [72].

Das Bakterium *R. solanacearum* ist einer der Modellorganismen unter den Pflanzenpathogenen und wurde bereits in vielen Studien biochemisch untersucht. Die GenBank-Annotation umfasst 3440 Gene auf dem Chromosom sowie 1676 auf dem Plasmid und beruht auf Vorhersagen mit dem speziell für diesen Organismus trainierten Programm FrameD [73] in Kombination mit komparativen Methoden. Für die Organismen *B. pseudomallei* und *R. solanacearum* wurde jeweils die komplette Annotation aus GenBank als Referenz verwendet.

4.5.2 Analyse des Verfahrens

Maß für die Performanz

Für die Evaluierung und den Vergleich der Performanz von Programmen werden generell die Parameter Sensitivität und Spezifität untersucht. Als Sensitivität wird die bereits

in Abschnitt 4.2.2 für die ROC-Analyse eingeführte TPR (*true positive rate*) bezeichnet, die zu diesem Zweck mit dem Faktor 100 skaliert wird. Der Wert wird für einen fixen, oft im Programm voreingestellten Schwellenwert berechnet. Der Begriff der Spezifität wird in der Bioinformatik gewöhnlicherweise anders verwendet als in anderen Bereichen der angewandten Statistik. In der angewandten Statistik wird allgemein der Wert FPR als Spezifität bezeichnet, während in Biologie/Bioinformatik der *positive Vorhersagewert* (*positive predictive value* – PPV), wiederum skaliert mit Faktor 100, verwendet wird. Wie für die Sensitivität wird dieser Wert für einen festgelegten Schwellenwert angegeben. Im Folgenden wird der Begriff Spezifität gleichbedeutend mit dem positiven Vorhersagewert verwendet.

Definition von Sensitivität (=TPR·100) und Spezifität (=PPV·100):

$$\text{TPR} = \frac{\#\text{TP}}{\#\text{TP} + \#\text{FN}} \tag{4.8}$$

$$\text{PPV} = \frac{\#\text{TP}}{\#\text{TP} + \#\text{FP}}, \tag{4.9}$$

In der Genannotation werden Sensitivität und Spezifität normalerweise auf die korrekte Vorhersage des Genstopps bezogen. Ein Gen wird also als korrekt vorhergesagt (TP) gewertet, wenn der vorhergesagte Stopp mit der Vergleichsannotation übereinstimmt. Die korrekte Vorhersage des Translationsstarts wird hierbei nicht einbezogen. Da in der vorliegenden Arbeit die Vorhersage von Translationsstarts untersucht wird, beziehen sich im Folgenden die Angaben TPR und PPV immer auf die Vorhersage des Translationsstarts. Zur Verdeutlichung wird der Begriff der TIS-Genauigkeit anstelle des Begriffs Sensitivität verwendet, wenn sich die Angabe sich auf den Translationsstart bezieht.

Die Performanz von TICO wurde hinsichtlich der TIS-Genauigkeit bei der Vorhersage auf verschieden Genomen untersucht. Dabei wurde zum einen der Einfluss der Glättung auf die Genauigkeit der Vorhersage analysiert (Abschnitt 4.5.3, 69), zum anderen die Performanz im Vergleich mit anderen Postprozessoren (4.5.5, S. 73). Außerdem wurde der Algorithmus hinsichtlich der Aussagekraft der berechneten PWM-Scores untersucht. Es wurden dazu PRC (*precision recall curves*) für alle Datensätze analysiert (siehe folgenden Abschnitt) und die Trennbarkeit der Score-Verteilungen der Klassen *weak* und *strong* betrachtet (siehe Anhang, B.1, S. 88).

Precision Recall Curves (PRC) - Analyse

Die PRC-Analyse ist ein Verfahren, das der bereits beschrieben ROC-Analyse (4.2.2) ähnelt [74]. Im Gegensatz zur ROC, wo TPR (*true positive rate*) und FPR (*false positive rate*) für alle

Schwellenwerte gegeneinander aufgetragen werden, wo also die Wahr-Negativen in die Bewertung eingehen, verwendet man bei der PRC-Analyse TPR und PPV (4.9). Die Analyse ist neben der Angabe von Sensitivität und Spezifität für einen fixen Schwellenwert dazu geeignet, die Performanz und Robustheit eines Algorithmus zu belegen. Davis und Goadrich [74] haben gezeigt, dass die PRC im Vergleich zur ROC oft die aussagekräftigeren Resultate liefert. Hinzu kommt, dass für viele bioinformatische Fragestellungen nicht klar definiert ist, welche Daten als Grundgesamtheit der Wahr-Negative zu betrachten sind. Daher wurde in dieser Untersuchung die TIS-Vorhersageperformanz von TICO mit Hilfe der PRC bewertet.

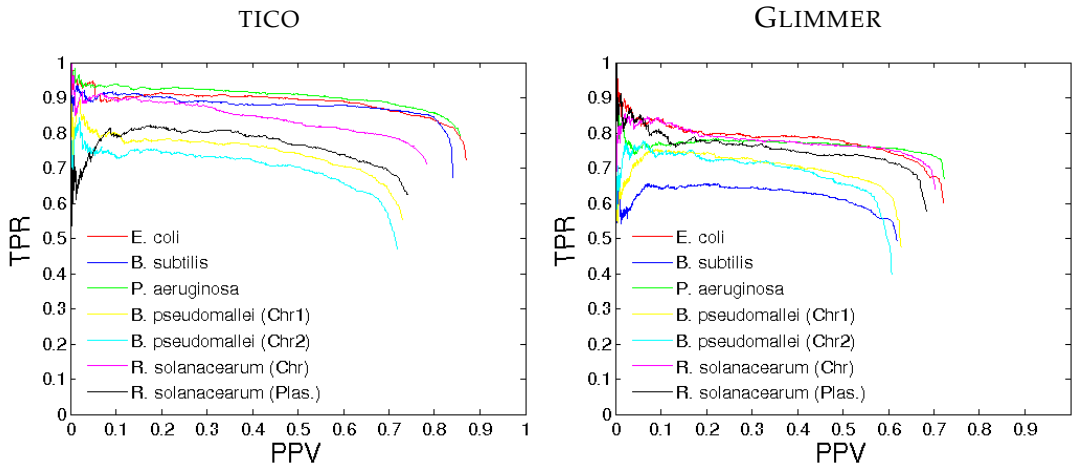

Abbildung 4.8: **Abbildung der PRC-Kurven für TICO-Vorhersagen und GLIMMER3-Vorhersagen. GLIMMER3 wurde in diesem Fall verwendet, da in der Ausgabe ein probabilistischer Score geliefert wird, der in der Ausgabe der älteren Version fehlt. Um die Vergleichbarkeit zu gewährleisten, wurden die entsprechenden GLIMMER3-Vorhersagen als initiale TIS-Annotationen für TICO verwendet.**

Wie bei der ROC-Analyse wird bei der PRC-Analyse das AUC-Kriterium (*area under curve*-Kriterium) als absolutes Maß der Performanz betrachtet. Da bei der Analyse ein aussagekräftiger Score für jede einzelne Vorhersage benötigt wird, konnte in diesem Fall nur ein Vergleich mit dem Programm GLIMMER3 durchgeführt werden. Die übrigen zur Evaluation herangezogenen Programme liefern keine Scores zu den einzelnen Vorhersagen.

Eine graphische Darstellung der Ergebnisse ist in Abbildung 4.8 gegeben. Für die Analyse wurden jeweils die Werte TPR (4.8) und PPV (4.9) verwendet. Die AUC-Werte sind in Tabelle 4.3 gegeben. Die Grafik zeigt, dass die PRC aller Datensätze für beide Programme einen nahezu waagerechten Verlauf haben. In den meisten Fällen liegen die für TICO ermittelten Kurven durchgängig über denen von GLIMMER, nur für den Orga-

nismus *B. pseudomallei* liegen die GLIMMER-Werte teilweise höher. Die PRC-Kurven für GLIMMER brechen in allen Fällen früher ab, wodurch TICO deutlich höhere AUC-Werte erreicht. Tabelle 4.3 zeigt die Überlegenheit von TICO hinsichtlich der Genauigkeit der TIS-Vorhersage gegenüber dem konventionellen Genvorhersageprogramm GLIMMER.

Tabelle 4.3: **AUC-Werte zu den PRC (*precision recall curve*) aus Abbildung 4.8. Für jede Eingabesequenz sind die AUC-Werte von TICO und GLIMMER3 angegeben. Die GLIMMER-Vorhersage wurde als initiale TIS-Annotation für TICO verwendet.**

	AUC-Kriterium	
Organismus	TICO	GLIMMER
E. coli K-12	0,772	0,565
B. subtilis	0,740	0,395
P. aeruginosa PAO1	0,776	0,555
B. pseudomallei K96243 Chr. 1	0,541	0,441
B. pseudomallei K96243 Chr. 2	0,508	0,423
R. solanacearum Chr.	0,663	0,548
R. solanacearum Plasmid	0,568	0,518

4.5.3 Verbesserung der Vorhersage von Translationsstarts mit TICO

Bei der Untersuchung der Performanz von TICO wurde als initiale TIS-Annotation die Vorhersage des weit verbreiteten Genvorhersageprogramms GLIMMER2.02 [10] verwendet. Sowohl GLIMMER als auch TICO wurde mit den Standardeinstellungen ausgeführt. Die Einstellung des Glättungsparameters wurde durch die automatische Adaption vorgenommen. Bei dieser Untersuchung wurde nicht die neueste GLIMMER-Version (GLIMMER3) eingesetzt, sondern Version GLIMMER2.02, da diese in vielen Genomen einen etwas höheren Anteil annotierter Gene vorhersagt. Somit ist die Gefahr geringer, Gene komplett zu übersehen. Dies ist ein Grund dafür, dass GLIMMER2.02 in der Genomannotation derzeit verbreiteter ist als GLIMMER3. Voruntersuchungen zeigten außerdem, dass GLIMMER2.02 eine etwas schlechtere TIS-Genauigkeit zeigt als GLIMMER3, wodurch die Grenzen der untersuchten Verfahren deutlicher aufgezeigt werden.

Für die untersuchten Genome sagt GLIMMER2.02 97,0% (Plasmid von *R. solanacearum*) bis 99,3% (*E. coli*) der Gene der entsprechenden Referenzannotation vorher. Die Genauigkeit der Vorhersage der Translationsstarts (TIS-Genauigkeit) ist im Vergleich dazu deutlich schlechter: Sie liegt um mehr als 30 Prozentpunkte unter dem Anteil gefundener Gene, im Bereich von 50% bis 60%. Die besten Ergebnisse erreicht GLIMMER für *E. coli*, wo das Programm 63,2% der verifizierten Translationsstarts aus dem EcoGene-Datensatz

korrekt vorhersagt. Durch TICO kann die Vorhersage in diesem Fall um 31 Prozentpunkte auf einen Wert von 94,2% verbessert werden. Für *B. subtilis* wird eine Verbesserung der Vorhersagegenauigkeit um 28,1 Prozentpunkte erreicht. Die Verbesserung der Vorhersagen für Genome mit hohem GC-Gehalt ist etwas geringer: Für *P. aeruginosa* beträgt sie 27,4 Prozentpunkte, für *R. solanacearum* 23,4- (Chromosom) bzw. 21,2 Prozentpunkte (Plasmid) und für *B. pseudomallei* 16,4- (Chromosom 1) bzw. 18,1 Prozentpunkte (siehe Tabelle 4.4, S. 69).

Die Ergebnisse zeigen deutlich, dass das Programm TICO sehr gut zur Reannotation von Translationsstarts anwendbar ist. Da keine sequenzspezifischen Informationen einbezogen werden, das Verfahren also vollständig unüberwacht arbeitet, kann eine gute Performanz auch erreicht werden, wenn keine sequenzspezifischen Charakteristika bekannt sind, wie es bei neusequenzierten Genomen häufig der Fall ist. Die Anwendung zur Reannotation von Genomen, deren Annotationen bereits in öffentlichen Datenbanken verfügbar sind, liegt ebenfalls nahe. Wie bereits gezeigt, basiert ein großer Teil dieser Annotationen auf automatischen Vorhersagen und ist vor allem hinsichtlich der Annotation der Translationsstarts stark fehlerbehaftet. Verglichen mit der verifizierten Annotation im EcoGene-Datensatz weist die GBK-Annotation des verhältnismässig gut annotierten *E. coli*-Genoms 10,2% falsch annotierte Translationsstarts auf. Mit Hilfe von TICO kann der Anteil falsch annotierter Translationsstart in diesem Fall auf 4,9% gesenkt werden.

Tabelle 4.4: **Übersicht der Verbesserungen der TIS-Genauigkeit bei der Vorhersage durch TICO. Angegeben ist jeweils die TIS-Genauigkeit von GLIMMER2, die entsprechende TIS-Genauigkeit von TICO und die Verbesserung in Prozentpunkten.**

	TIS-Genauigkeit		Verbesserung
Organismus	GLIMMER	TICO	(Prozentpunkte)
E. coli K-12 (EcoGene)	63,2	94,2	31,0
B. subtilis (*non-y*)	61,3	89,4	28,1
PseudCAP	57,8	84,8	27,4
B. pseudomallei K96243 Chr. 1	53,2	69,4	16,4
B. pseudomallei K96243 Chr. 2	48,9	67,0	18,1
R. solanacearum Chr.	51,5	74,7	23,4
R. solanacearum Plasmid	48,9	70,1	21,2

Einfluss der Glättung auf die Vorhersageperformanz

Neben der automatischen Einstellung der Glättung bietet TICO die Möglichkeit einen fixen Wert für den Glättungsparameter σ vorzugeben. Um den Einfluss des Glättungsparameters auf die Performanz zu untersuchen, wurde die Vorhersagegenauigkeit für fixe

$\sigma \in \{0{,}1, \ldots, 2{,}0\}$ betrachtet. Im Bereich $\sigma \in \{0{,}4, \ldots, 1{,}7\}$ bleibt die Vorhersage für alle Genome stabil und weicht für *E. coli* und *B. subtilis* um maximal 2 und im Fall der Genome mit hohem GC-Gehalt um maximal 4 Prozentpunkte vom besten Wert ab. Für *E. coli* und *B. subtilis* bleiben die Werte auch darüber hinaus stabil, erst bei $\sigma = 1{,}9$ verschlechtert sich die Vorhersageperformanz signifikant. Der stärkste Abfall der Performanz ist für *P. aeruginosa* bei $\sigma \geq 1{,}7$ zu beobachten. Abbildung 4.9 zeigt die Abtastung der TIS-Genauigkeit für alle Genome bei den σ-Werten im untersuchten Bereich. Die besten Werte jeweils für die automatische Adaption und fixen Glättungsfaktor sind in der Tabelle 4.5 zusammengefasst. Alle ermittelten Werte sind in der Tabelle »Sigma-Sampling« B.2 im Anhang (S. 86) aufgelistet.

Tabelle 4.5: **Gegenüberstellung der Performanz mit automatischer Adaption des Glättungsfaktors σ und der »bestmöglichen« Performanz für einen fixen Gättungswert.**

Organismus	**automatische Adaption**		**fixer Glättungsfaktor**	
	Performanz	ermitteltes σ	Performanz	vorgegebenes σ
E. coli K-12	94,2	0,45	**94,3**	0,5
B. subtilis	89,4	0,45	89,4	0,45
P. aeruginosa PAO1	84,8	0,55	**85,2**	0,6
B. pseudomallei K96243 Chr. 1	69,4	0,5	**69,8**	0,6
B. pseudomallei K96243 Chr. 2	67,0	0,5	**67,2**	0,6
R. solanacearum Chr.	74,7	0,5	**75,2**	0,6
R. solanacearum Plasmid	70,1	0,45	**71,0**	0,6

Es zeigt sich, dass die automatische Adaption für die untersuchten Genome keine signifikante Verschlechterung der Vorhersagegenauigkeit gegenüber dem optimalen Wert bewirkt. In Tabelle 4.5 sind die Werte der automatischen Adaption des Glättungsparameters den optimalen Werten gegenübergestellt. Im Vergleich zum besten Wert von 94,3% bei $\sigma = 0{,}5$ erreicht TICO für *E. coli* mit automatischer Adaption 94,2% ($\sigma = 0{,}45$), für *B. subtilis* sind beide Werte identisch: 89,4% bei $\sigma = 0{,}45$. Für *P. aeruginosa* liegt die TIS-Genauigkeit 0,4 Prozentpunkte unter dem optimalen Wert bei $\sigma = 0{,}6$, ebenso wie für *B. pseudomallei* Chr. 1. Für Chr. 2 ist der Wert nur um 0,2 Prozentpunkte schlechter. Die maximale Abweichung wurde für *R. solanacearum* festgestellt, wo die TIS-Genauigkeit durch die automatische Einstellung der Glättung um 0,5- (Chr.) bzw. 0,9 Prozentpunkte sinkt. Für die untersuchten Genome beträgt die Verschlechterung durch die automatische Selektion des Glättungsfaktors im ungünstigsten Fall also 0,9 Prozentpunkte. Da das Festsetzen des Glättungsfaktors durch den Benutzer einen gewissen Grad an Hintergrundwissen erfordert, ist die automatische Adaption als Standardeinstellung vorgegeben.

Die Untersuchung der Performanz von TICO für fixe σ und der Vergleich mit der Performanz für die automatische Selektion, wie sie hier beschrieben wird, ist in der Praxis im Allgemeinen nicht möglich, da es in der Regel keine verlässliche Referenz gibt, mit der die Vorhersage verglichen werden kann. Die Untersuchung diente vor allem dazu, die oberen und unteren Grenzen des Algorithmus auszuloten. Da die automatische Glättung als Standardeinstellung vorgegeben ist und dem Benutzer ausdrücklich empfohlen wird, wird im Weiteren ausschließlich die Performanz für die automatische Glättung angegeben.

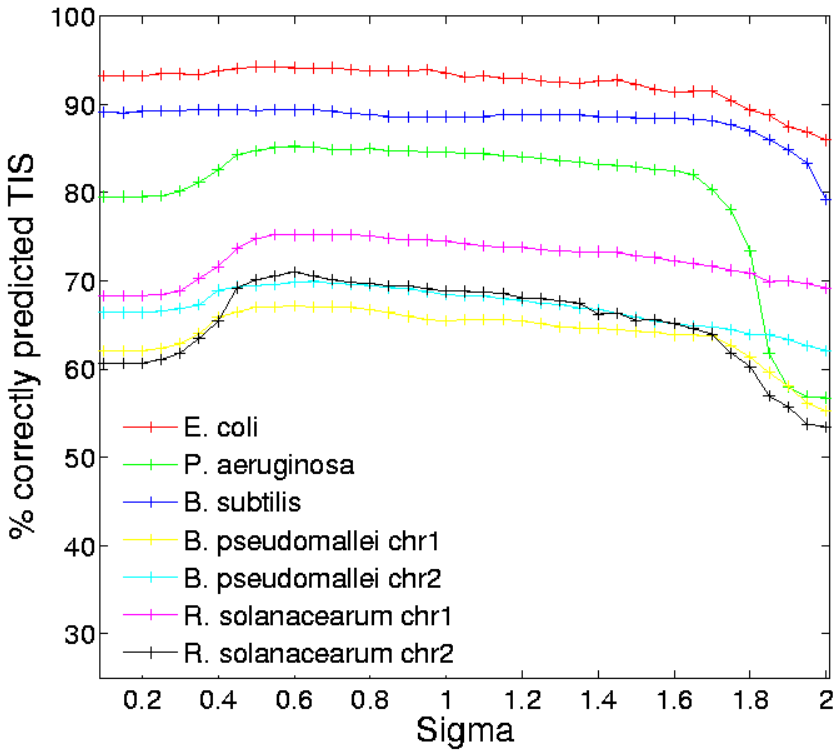

Abbildung 4.9: **Performanz bei der Vorhersage für Glättung $\sigma \in \{0{,}1, \ldots, 2{,}0\}$. Aufgetragen sind die korrekt vorhergesagten TIS in % (Ordinate) gegen die abgetasteten σ (Abszisse). Im Bereich $\sigma \approx 0{,}4$ bis $\sigma \approx 1{,}8$ ist die Vorhersageperformanz für alle Genome stabil. Für *E. coli* und *B. subtilis* ist sie auch darüber hinaus für $\sigma < 0{,}4$ noch stabil, für die Genome mit hohem GC-Gehalt sinkt die Performanz bei $\sigma \leq 0{,}4$ und $\sigma \geq 1{,}7$ etwas.**

4.5.4 Vergleich des Algorithmus mit anderen Methoden

TICO wurde hinsichtlich seiner Methodik und Vorhersagequalität mit folgenden *state-of-the-art*-Programmen für die Reannotation von prokaryotischen Genstarts verglichen:

GS-Finder [12], MED-Start [13] und RBS-Finder [62]. Diese wurden bereits 2004 in einer Veröffentlichung von Zhu *et al.* [13] einander gegenübergestellt. Zhu *et al.* zeigen, dass die Programme auf den verifizierten Datensätzen der Standardorganismen *E. coli* und *B. subtilis* eine gute Performanz aufweisen.

Es ist zu berücksichtigen, dass die von Zhu *et al.* betrachteten Programme im Gegensatz zu TICO sequenzspezifische Parameter bei der Vorhersage einbeziehen. Diese Parameter werden in der Regel auf Modellorganismen wie *E. coli* voreingestellt und getestet. Es ist jedoch nicht absehbar, welche Vorhersageperformanz diese Programme auf Genomen zeigen, deren Charakteristika von denen der Modellorganismen abweichen. Daher werden zunächst die Methodiken der Programme diskutiert, insbesondere hinsichtlich voreingestellter Werte, eingegangen. Die Vorhersageperformanz der Programme auf verschiedenen Genomen wird im folgenden Abschnitt angegeben und verglichen.

RBS-Finder (Suzek *et al.*, 2001)

Sofern keine RBS-Motive vorgegeben werden, schätzt das Programm die Muster für potentielle Ribosombindestellen mit der Tompa-Methode [60], einem probabilistischen Verfahren. Basierend auf den geschätzten (oder vorgegebenen) Motiven wird ein Score für die möglichen RBS-Sequenzen upstream potentieller Translationsstarts berechnet. Dieser geht in einen Gesamt-Score ein, der für jeden Kandidaten berechnet wird. Ist der ermittelte Gesamt-Score eines Startkandidaten höher als der des annotierten Starts aus der Eingabe, so erfolgt eine Reannotation, sofern dadurch keine Überlappungen von mehr als 30BP entstehen.

Der Gesamt-Scores berechnet sich aus dem RBS-Score, aus einer Bewertung desjenigen Codons, welches den potentiellen Translationsstart bezeichnet (ATG, GTG, TTG), und aus der Distanz des Kandidaten zum annotierten Start ein. Die erwarteten Häufigkeiten der Codons ATG, GTG und TTG als Translationsstarts, sowie die erwartete Distanz reannotierter Starts zum Start in der initialen Annotation und die Richtung der Verschiebung (upstream oder downstream) sind dabei als empirische Werte sind vorgegeben.

GS-Finder (Ou *et al.*, 2004)

GS-Finder basiert wie TICO auf einer iterativen Prozedur, bei der in jeder Interation ein probabilistisches TIS-Modell geschätzt wird. Mit dem Modell werden die einzelnen TIS-Kandidaten bewertet. Gemäß dieser Bewertung werden in jedem Iterationsschritt die vorhergesagten TIS-Positionen angepasst, die als Basis für den nächsten Schritt dient.

Das Modell setzt sich aus sechs Komponenten zusammen von denen drei die Nukleotid-Verteilung in der Region um potentielle Translationsstarts modellieren. Die anderen drei Komponenten modellieren das *Coding*-Potential, die Wahrscheinlichkeit des Startcodons und den Abstand zu dem am weitesten upstream gelegenen Startkandidaten (*leftmost*). Die Berechnung der Nukleotidverteilungen um potentielle Translationsstarts und des Coding-Potentials basieren auf der *Z-curve*-Repräsentation der Sequenz [75]. Die Nukleotidverteilungen werden ausschließlich aus den Eingabedaten geschätzt. Das Coding-Potential wird über einen Z-Score berechnet. Der Schwellenwert für die Unterscheidung von kodierend und nicht-kodierend ist fix auf den Wert 0 gesetzt, so dass Regionen mit einem positiven Z-Score als kodierend, Regionen mit einem negativen Wert als nicht-kodierend bewertet werden. Die erwartete Startcodon-Usage und der erwartete Abstand zwischen dem am weitesten upstream gelegenen TIS-Kandidaten und *wahrer* TIS sind mit empirischen Werten voreingestellt.

MED-Start (Zhu *et al.*, 2004)

Das Programm MED-Start berechnet ebenso wie TICO und GS-Finder in einer iterativen Prozedur ein probabilistisches TIS-Modell. Für jeden TIS-Kandidaten wird in jeder Iteration ein Score berechnet, der zur Klassifikation als *wahre* oder *falsche* TIS dient. Das Modell setzt sich aus vier Komponenten zusammen: Dem potentiellen RBS-Motiv, dem *Spacer* (Distanz in BP) zwischen einer potentiellen RBS und der zugehörigen TIS, dem Sequenzkontext um potentielle TIS im Bereich -4...+15 BP und der Position des Kandidaten relativ zu den anderen Kandidaten des selben ORFs.

Die meisten der von MED-Start verwendeten Parameter, werden unüberwacht aus den Eingabedaten geschätzt. Vorgegeben ist jedoch die Länge der zu schätzenden SD-Sequenzen (Standardwert 5 BP) und eine maximale Anzahl von Motiven, die hierbei einbezogen werden (5). Aus der Beschreibung des Algorithmus in [13] geht nicht hervor, wie die Anzahl der Motive festgelegt wird, die tatsächlich bei einer Vorhersage einbezogen werden. Weiterhin geht mit Bezugnahme auf Ou *et al.* die Annahme ein, dass derjenige Kandidat eines ORFs, der am weitesten upstream liegt (*leftmost*), mit der höchsten Wahrscheinlichkeit der *wahre* Translationsstart ist, so dass das Programm – wie die klassischen Genvorhersageprogramme – eine Tendenz zur Maximierung der ORF-Länge zeigt.

4.5.5 Vergleich der Performanz mit anderen Postprozessoren

Bei allen Vergleichen wurden sowohl TICO als auch die anderen Programme mit ihren Standardparametern verwendet. Für jedes der getesteten Programme wurde die Vorher-

sage des Programms GLIMMER2.02 [10] als Eingabe verwendet. Gemessen wurde jeweils der Anteil korrekt vorhergesagter Translationsstarts im Vergleich zur Referenzannotation. Die Referenzannotationen sind im Detail unter 4.5.1, S. 64 beschrieben. Die Sensitivität von GLIMMER (in % gefundener ORFs) sowie die TIS-Genauigkeit der Programme sind in Tabelle 4.6 aufgeführt.

Auf den Genomen von *E. coli* und *B. subtilis* zeigen TICO, MED-Start und GS-Finder eine vergleichbar gute Performanz. Der Anteil korrekt vorhergesagter TIS differiert in diesem Fall um maximal 3,9 Prozentpunkte. Die GLIMMER-Vorhersage der TIS-Position kann durch die Postprozessoren um mindestens 26,6 Prozentpunkte (GS-Finder) verbessert werden. TICO zeigt hier mit 94,2% (*E. coli*) bzw. 89,4% (*B. subtilis*) die beste Performanz (Verbesserung um 31,0- bzw. 28,1 Prozentpunkte). Die Raten für RBS-Finder sind um etwa 10 Prozentpunkte schlechter als die der anderen Postprozessoren.

Tabelle 4.6: **Performanz des TICO-Algorithmus im Vergleich zur initialen Annotation durch GLIMMER und zu den Postprozessoren RBSfinder [62], GS-Finder [12] und MED-Start [13]. Angegeben ist jeweils der prozentuale Anteil korrekt vorhergesagter Translationsstarts (TIS-Sensitivität), verglichen mit dem Referezdatensatz. Für GLIMMER wurde außerdem die Sensitivität auf ORF-Level (% gefundene ORFs) angegeben. Eine nähere Beschreibung der Datensätze ist in Abschnitt 4.5.1 gegeben.**

Daten	Anzahl d. Gene	% gefundene ORFs	% korrekt vorhergesagte TIS				
			GLIMMER	MED-Start	TICO	GS-Finder	RBSfinder
EcoGene	854	99,3	63,2	92,0	**94,2**	90,3	81,9
B. subtilis non-y	1248	98,6	61,3	89,2	**89,4**	87,9	78,5
PseudoCAP	3281	97,5	57,8	3,6	**85,2**	83,6	67,7
B. pseudom. Chr. 1	3399	97,7	53,2	5,5	**69,6**	64,3	53,3
B. pseudom. Chr. 2	2329	97,7	48,9	4,7	67,0	**67,5**	52,1
R. solanac. Chr.	3440	97,2	51,5	5,0	**74,9**	71,4	56,8
R. solanac. Plasmid	1676	97,0	48,9	6,0	**70,1**	66,2	55,5

Die Raten für Genome mit hohem GC-Gehalt sind generell niedriger als die für Genome mit moderatem GC-Gehalt. TICO und GS-Finder zeigen hier vergleichbare Performanz. Sie unterscheidet sich auf den getesteten Genomen mit hohem GC-Gehalt um maximal 5,3 Prozentpunkte. Die besten Werte erreichen die Programme auf dem gut annotierten PseudoCAP-Datensatz, wo die TIS-Genauigkeit bei 85,2% (TICO) und 83,5% (GS-Finder) liegt. Die jeweils maximale TIS-Genauigkeit auf den weiteren Genomen mit hohem GC-Gehalt ist bei *B. pseudomallei* 69,6% für Chr. 1 (TICO) und 67,5% für Chr. 2 (GS-Finder), sowie bei *R. solanacearum* 74,9% für Chr. (TICO) und für das Plasmid 70,1% (TICO).

Die Raten von RBS-Finder liegen um 15 bis 20 Prozentpunkte niedriger. MED-Start versagt bei der Vorhersage auf allen getesteten Genomen mit hohem GC-Gehalt. Die maximal erreichte Rate liegt bei 6,0%, die schlechteste bei 3,6%. Um zu klären, wie diese schlechten Raten zustande kommen, wurden die von MED-Start als SD-Motive vorhergesagten Muster untersucht.

Für *P. aeruginosa* wurden von MED-Start folgende fünf Motive als SD-Motiv vorhergesagt: $CCTGG$, $GCGCC$, $GCCTG$, $CGCCG$ und $CGGCG$. Diese Muster unterscheiden sich stark von bislang bekannten SD-Motiven, welche für bakterielle Genome generell als vier oder mehr aufeinander folgende Basen der Form des Regulären Ausdrucks $RGGRGGTGAT$ ($R = A$ oder G) beschrieben werden [45]. Der Vergleich dieser Muster mit den von TICO gelernten PWMs (*positional weights matrices*) zeigt, dass den Trimeren, aus denen sich diese Muster zusammensetzen (wie CCT, TGG, ...), in der upstream-Region von TIS fast ausschließlich hohe negative Gewichte zugeordnet werden. In Abbildung 4.10 (S. 76) sind exemplarisch die TICO-Gewichte der Trimere CCT, GCC, CGC und GCG aufgetragen. Diese Trimere treten am häufigsten als Teilwort der von MED-Start vorgeschlagenen SD-Motive für *P. aeruginosa* auf. Vor allem in der Region -12 bis -8, in der eine SD-Sequenz zu vermuten wäre, werden diese Trimere von TICO als Hinweis für eine »nicht«-TIS betrachtet. MED-Start hat hier offensichtlich Muster von Motiven gelernt, die häufig in der upstream-Region von TIS-Kandidaten auftreten, die aber keineswegs im Zusammenhang mit charakteristischen Signalen von wahren TIS stehen.

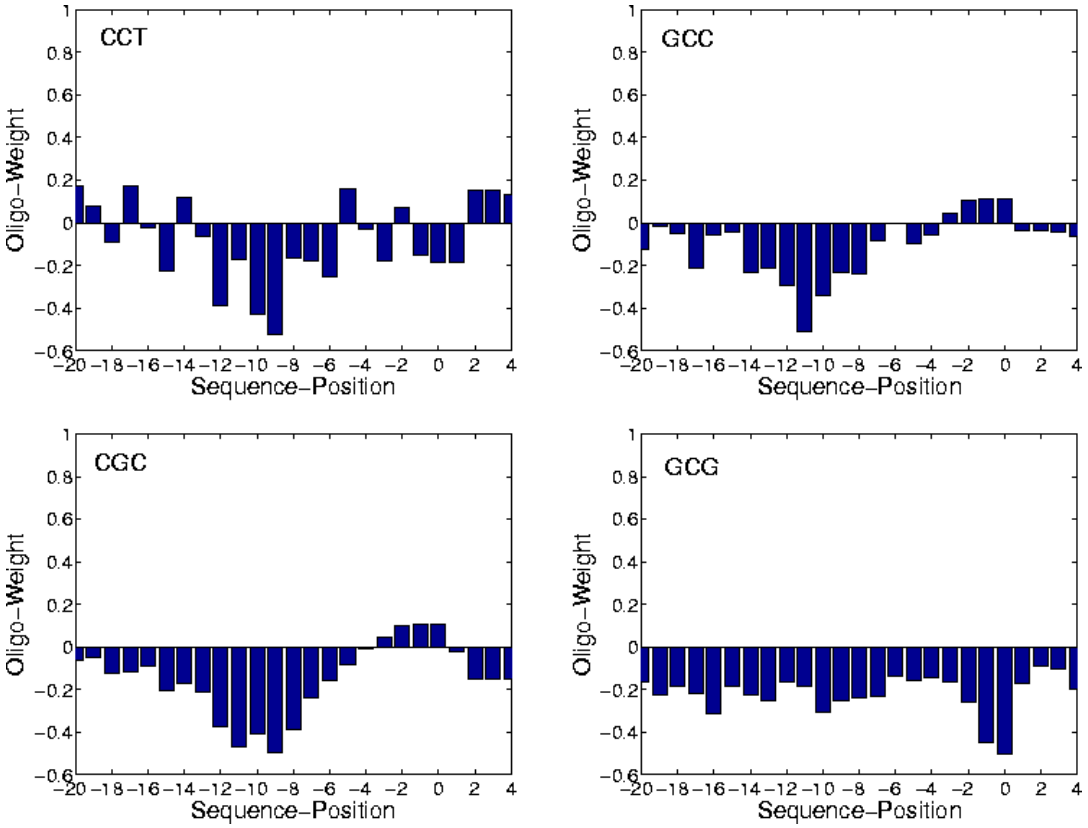

Abbildung 4.10: **In der Abbildung sind exemplarisch von TICO berechnete Trimer-Gewichte für die upstream-Region von TIS-Kandidaten dargestellt. Die TIS-Kandidaten sind jeweils an Position 0 angegeben, die upstream-Positionen tragen negatives Vorzeichen. Negative Gewichte gehen bei der Klassifikation der Kandidaten durch TICO als Hinweis für eine falsche TIS ein. Die gezeigten Trimere CCT, GCC, CGC und GCG treten als häufigste Teilworte in den von MED-Start vorgeschlagenen SD-Motiven für *P. aeruginosa* auf. Vor allem in der Region -12 bis -8 sind diese Trimere durchweg mit stark negativen Gewichten assoziiert.**

Kapitel 5

Schlussfolgerungen

Im Rahmen dieser Arbeit wurden zwei Ansätze zur Verbesserung der Vorhersage von Translationsstarts in prokaryotischen Genomen entwickelt: Ein überwachtes Verfahren, der Oligo-Kern-Algorithmus [1], und ein unüberwachtes Clusterverfahren, das in dem Programm TICO [2, 3, 4] implementiert wurde.

Mit dem Oligo-Kern-Algorithmus wird ein neues Konzept zum Datamining auf biologischen Sequenzen vorgestellt. Es konnte gezeigt werden, dass das Verfahren eine hohe Performanz bei der Klassifikation und der Detektion von biologisch relevanten Signalen aufweist. Durch die intuitive Visualisierung der gelernten Charakteristika ist das Verfahren gut zum Datamining geeignet. Es kann bei der Suche nach unbekannten Signalen eingesetzt werden und Hinweise auf die tatsächliche Relevanz bekannter Signale geben. Bei diesem Ansatz wird außerdem die Glättung von Oligomer-Vorkommen als neues Konzept vorgestellt und erfolgreich angewendet. Die Performanz des Algorithmus wurde am Fallbeispiel prokaryotischer Translationsstarts gezeigt. Signale zur Initiation der Translation bei dem untersuchten Organismus *E. coli* konnten eindeutig und korrekt mit der ihnen innewohnenden Variabilität detektiert werden.

Es wird weiterhin gezeigt, dass eine Steigerung dieser Performanz durch eine additive Kombination mehrerer Oligo-Kerne (*kombinierte Oligo-Kerne*) erzielt werden kann. Eine Kombination mit anderen Kernen zur Steigerung der Performanz ist ebenfalls denkbar. Der Oligo-Kern-Algorithmus kann durch die Flexibilität hinsichtlich der Positionsabhängigkeit zur Suche anderer biologischer Signale angepasst werden. Das Verfahren wurde bereits bei der Suche nach Spleißstellen (*splice sites*) verwendet [55]. Der Algorithmus wurde zwecks der Optimierung der Glättung von Igel *et al.* weiterentwickelt.

Das Programm TICO (*Translation Initiation site COrrection*), das zweite in dieser Arbeit vorgestellte Verfahren, ermöglicht die Reannotation von Genstarts mittels eines unüber-

wachten Lernschemas. Das erfolgreiche Konzept der Glättung von Oligomer-Vorkommen wird auch bei diesem Verfahren eingesetzt. Das Programm zeigt eine sehr gute Performanz bei der Reannotation von Translationsstarts. Auf den getesteten Genomen wird eine Verbesserung der Vorhersage um bis zu 30 Prozentpunkte gegenüber klassischen Genvorhersageprogrammen erreicht. Der Vergleich mit anderen *state-of-the-art* Programmen zeigt, dass die Vorhersageperformanz von TICO in den meisten Fällen besser ist als die anderer Postprozessoren, obwohl das Programm ohne sequenzspezifische Parameter arbeitet. Wie auch beim Oligo-Kern-Algorithmus ist die Visualisierung der Merkmale, die bei der Klassifikation einbezogen werden, möglich. Die Signale sind hier allerdings weniger deutlich, da das Signal-Rausch-Verhältnis schlechter ist, als bei dem überwachten Verfahren. Dennoch kann die Visualisierungsfunktion Hinweise auf die Güte der Eingabedaten und der Klassifikation geben. Starke Signale wie die SD-Sequenz werden auch bei diesem Verfahren durch die Visualisierung sichtbar.

Das Programm TICO wird auf Grund seiner Vorhersagegenauigkeit unter anderem in den Software Pipelines des Göttingen Genomics Laboratory (G_2L) [76] und des PathoSystems Resource Integration Center (PATRIC) [77] zur Reannotation von Translationsstarts verwendet. Das Programm wird außerdem vom Virginia Bioinformatics Institute als Web-Service angeboten [78].

In weiteren, bislang nicht veröffentlichten Studien wird untersucht, inwiefern sich die Gewichtsmatrizen, also die Merkmale der Translationsstarts eines Organismus, auf andere Organismen übertragen lassen. Solche Studien können darüber Aufschluss geben, ob bzw. welche Charakteristika der prokaryotischen Translationsinitiation speziesspezifisch oder speziensübergreifend sind. Sind relevante Signale speziesspezifisch, so können sie als »Fingerabdruck« eines Organismus bzw. einer Gruppe genutzt werden, beispielsweise bei der phylogenetischen Einordnung metagenomischer Fragmente. Sind sie speziesübergreifend, so können sie bei einer speziesübergreifenden Genvorhersage genutzt werden.

Anhang A

Ergänzende Daten zum Datamining bei Translationsstarts in *E. coli* K-12

A.1 Oligo-Kern-Gewichte der Mononukleotide

Die größte Relevanz bei der Klassifikation hat G; es tritt verstärkt im Bereich der SD-Sequenz (-12...-7 BP) auf, in anderen Bereichen des Sequenzausschnittes vermindert. In der upstream-Region bis etwa -20 BP und besonders downstream an den Positionen 1 und 10 deutet das Auftreten von A offenbar verstärkt auf einen Translationsstart hin (siehe auch Abbildung A.3). Das Maximum liegt an Position 0 und ist auf die überdurchschnittliche Repräsentation von ATG als Startcodon zurückzuführen. Damit sind auch die negativen Spitzen von G und T an dieser Position zu begründen. Auffällig ist, dass A im Bereich direkt vor dem Translationsstart (-3...-1 BP) leicht negative Gewichte aufweist. C dagegen hat ein Maximum an Position -1. T hat in der upstream-Region -25 BP bis -15 BP einen leicht positiven Einfluss. In der Region bis -10 BP upstream von Translationsstarts tritt T vermindert auf. Vor allem im Bereich der SD-Sequenz weisen sowohl T als auch C stark negative Gewichte auf.

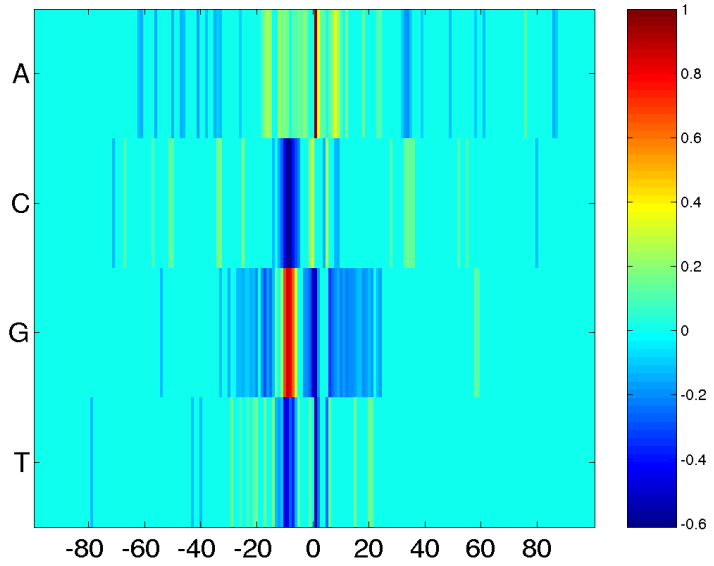

Abbildung A.1: **Darstellung der Gewichtsmatrix für alle Mononukleotide bei der Klassifikation von Translationsstarts in *E. coli* K-12.**

A.2 Oligo-Kern-Gewichte der Dinukleotide

Die hohen positiven Gewichte von GG, GA und AG lassen den Bereich der SD-Region (-12...-7 BP) erkennen. Vor allem GG hat in den restlichen Regionen um den Translationsstart (-3...20 BP) negative Gewichte (siehe auch Abbildung A.3). Das Maximum von AT an Position 0, sowie die Minima von GT und TT an dieser Position sind auf die Überrepräsentation von ATG als Startcodon zurückzuführen. Auffällig sind vor allem die relativ hohen positiven Gewichte von AA in der downstream-Region 1...10 BP und von TA in der upstream-Region -7...0 BP. An Position -1 haben TA und CA ein Maximum (siehe Abbildung A.3), was wiederum durch das häufige Auftreten von ATG als Translationsstart zu erklären ist. Offenbar treten T und C vermehrt an Position -1 auf, was sich auch bei den Oligomeren höherer Ordnung zeigt. Weiterhin treten die negativen Gewichte für GC und CA im Bereich der SD-Sequenz deutlich hervor.

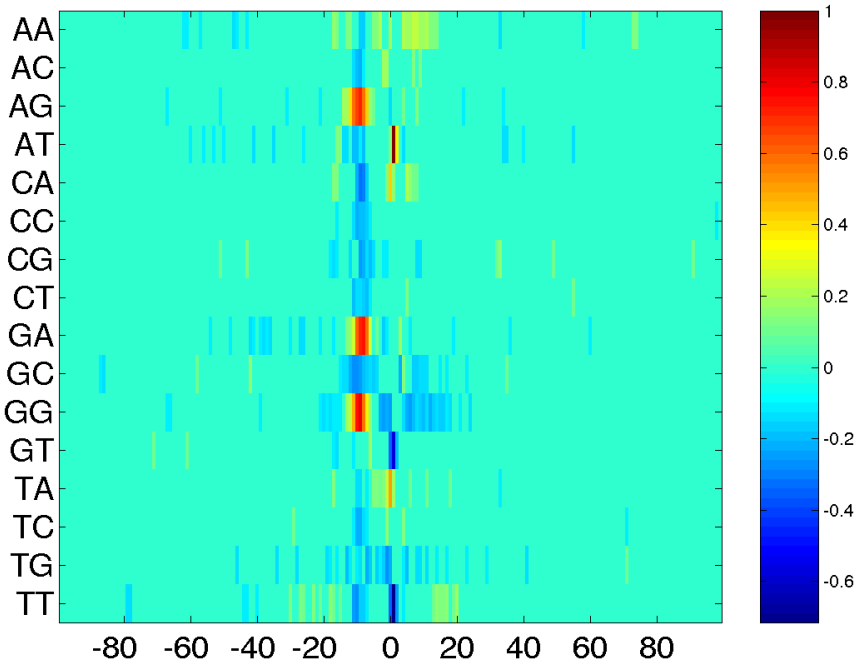

Abbildung A.2: **Darstellung der Gewichtsmatrix für alle Dinukleotide bei der Klassifikation von Translationsstarts in *E. coli* K-12.**

Tabelle A.1: **Peak-Positionen aller Dimer-Funktionen in alphabetischer Reihenfolge (AA,... TT). Angegeben sind jeweils Position und normalisierter Wert des Maximums/Minimums der entsprechenden Oligo-Funktion. Wie in den Abbildungen sind die Positionen bezüglich eines potentiellen Startcodons (Position 0) angegeben. Die Positionen mit negativem Vorzeichen bezeichnen den upstream-Bereich, bezogen auf einen TIS-Kandidaten.**

Nr.	Dimer	Peak-Position	Peak-Wert
1	AA	6	0,2475
2	AC	-10	-0,2157
3	AG	-10	0,7110
4	AT	0	1,0000
5	CA	-1	0,4144
6	CC	-11	-0,2114
7	CG	-10	-0,2324
8	CT	-8	-0,2039
9	GA	-9	0,7496
10	GC	-11	-0,2830
11	GG	-10	0,7889
12	GT	0	-0,5670
13	TA	-1	0,4938
14	TC	-10	-0,2185
15	TG	-2	-0,2800
16	TT	0	-0,7145

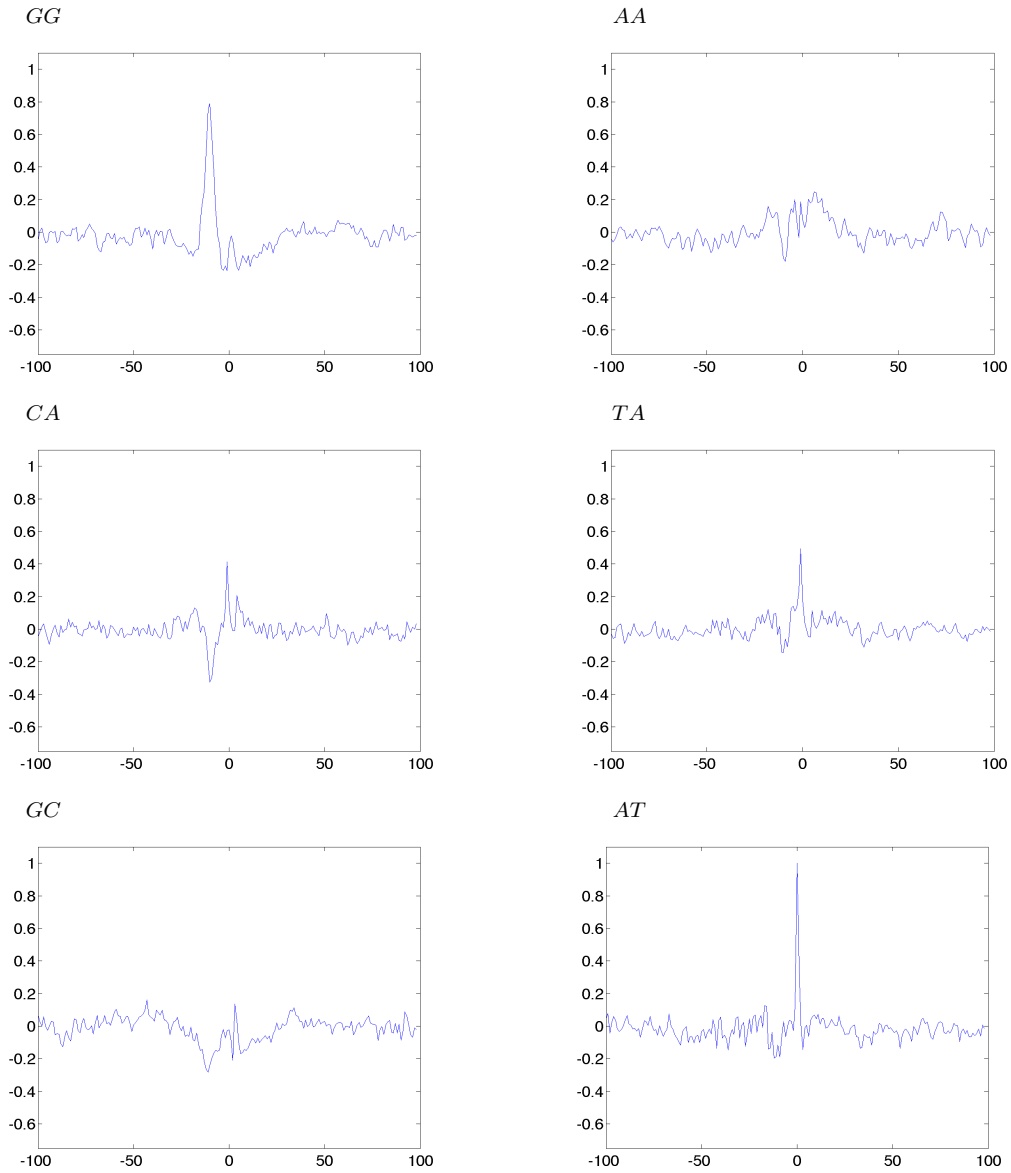

Abbildung A.3: **Exemplarische 2D-Darstellungen der Gewichtsfunktion von Dinukleotiden bei der Klassifikation von Translationsstarts in *E. coli* K-12. Die Werte sind so normiert, dass der maximale Wert aus alle Dimeren (AT an Position 0) 1 beträgt. Eine Beschreibung der Phänomene ist bei der Übersichtsdarstellung der Gewichte der Dimere gegeben.**

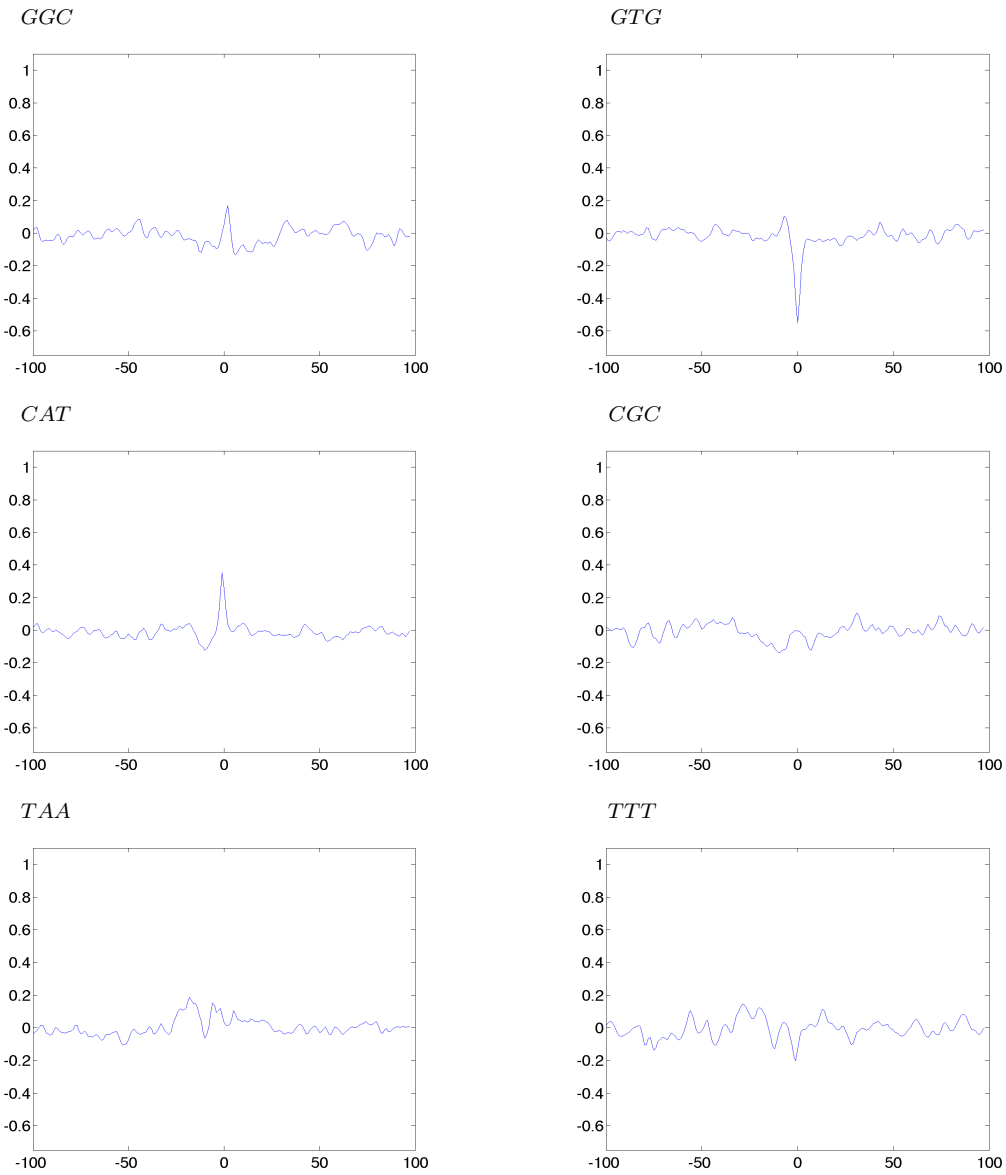

Abbildung A.4: **Exemplarische 2D-Darstellungen der Gewichtsfunktion von Trinukleotiden bei der Klassifikation von Translationsstarts in *E. coli* K-12. Die Werte sind so normiert, dass der maximale Wert (*ATG* an Position 0) 1 beträgt.**

Tabelle A.2: **Peak-Positionen der aller Trimer-Funktionen in alphabetischer Reihenfolge (*AAA*,... *TTT*). Angegeben sind jeweils Position und normalisierter Wert des Maximums/Minimums der entsprechenden Oligo-Funktion. Wie in den Abbildungen sind die Positionen bezglich eines potentiellen Startcodons (Position 0) angegeben. Die Positionen mit negativem Vorzeichen bezeichnen den upstream-Bereich, bezogen auf einen TIS-Kandidaten.**

Nr.	Trimer	Peak-Position	Peak-Wert	Nr.	Trimer	Peak-Position	Peak-Wert
1	AAA	9	0,1943	33	GAA	-8	0,1801
2	AAC	6	-0,1359	34	GAC	-13	0,0869
3	AAG	-12	0,3173	35	GAG	-10	0,8643
4	AAT	-18	0,1510	36	GAT	2	-0,1894
5	ACA	-2	0,1435	37	GCA	-11	-0,1365
6	ACC	-11	-0,1007	38	GCC	13	-0,1083
7	ACG	-10	-0,0954	39	GCG	7	-0,1578
8	ACT	-30	-0,0942	40	GCT	-12	-0,1342
9	AGA	-10	0,2961	41	GGA	-11	0,8832
10	AGC	-14	-0,0811	42	GGC	2	0,1692
11	AGG	-11	0,8124	43	GGG	-11	0,2701
12	AGT	-1	-0,1420	44	GGT	-1	-0,2580
13	ATA	-6	0,1210	45	GTA	-7	0,077
14	ATC	70	-0,1236	46	GTC	2	0,1400
15	ATG	0	1,0000	47	GTG	0	-0,5527
16	ATT	-1	-0,2440	48	GTT	-1	-0,2177
17	CAA	-10	-0,1957	49	TAA	-18	0,1867
18	CAC	-11	-0,1007	50	TAC	-11	-0,0842
19	CAG	-13	0,1534	51	TAG	31	0,0506
20	CAT	-1	0,3551	52	TAT	-1	0,4246
21	CCA	-11	-0,1210	53	TCA	-2	-0,1809
22	CCC	-87	-0,0877	54	TCC	-10	-0,0855
23	CCG	-8	-0,0866	55	TCG	-19	-0,1351
24	CCT	-9	-0,0954	56	TCT	29	-0,0662
25	CGA	47	0,0771	57	TGA	1	0,1353
26	CGC	-10	-0,1346	58	TGC	1	-0,3015
27	CGG	8	-0,1178	59	TGG	-3	-0,1634
28	CGT	-12	-0,1150	60	TGT	-1	-0,1695
29	CTA	-2	0,1789	61	TTA	-2	0,2207
30	CTC	55	0,0770	62	TTC	-11	-0,1142
31	CTG	-6	-0,1494	63	TTG	0	-0,6734
32	CTT	-12	-0,1084	64	TTT	-1	-0,2053

Anhang B

Ergänzende Daten zur Vorhersage von Translationsstarts mit TICO

B.1 Syntaxangabe zum GFF-Format nach der Spezifikation des Sanger Institutes

Das GFF-Format, das von TICO als Ausgabeformat angeboten wird, basiert auf der Spezifikation des Sanger Institutes [69] und kann mit dem Programm ARTEMIS angezeigt werden. Zeilenweise werden sogenannte *Features* dargestellt, die jeweils durch ein Schlüsselwort gekennzeichnet sind, beispielsweise steht das Schlüsselwort `CDS` für *coding region*. Die einzelnen Einträge sind durch Tabulatoren getrennt. Fehlende Einträge werden durch einen Punkt gekennzeichnet. Eine vollständige Beschreibung des GFF-Formates und aller *Features* ist unter
http://www.sanger.ac.uk/Software/formats/GFF/GFF_Spec.shtml
zu finden. Das Format ist für die TICO-Ausgabe um das Feature `REANNCDS` erweitert worden. Das Format setzt sich in folgender Weise aus den in Tabelle B.1 beschriebenen Einträgen zusammen:

```
<seqname> <source> <feature> <start> <end> <score> <strand> <frame> [attributes] [comments]
```

Tabelle B.1: **Erklärung der Einträge in der Syntaxangabe. Die Beispiele sind dem Ausgabebeispiel (4.3.3, S. 60) entnommen.**

Eintrag	Erläuterung	Beispiel
`seqname`	Name der Sequenz	`NC_006350`
`source`	Herkunft der Information	`Glimmer/tico`
`feature`	Merkmal, auf das sich der Eintrag bezieht	`REANNCDS`
`start`	Startposition des Merkmals	`1`
`end`	Endposition des Merkmals	`1116`
`score`	Score	`22.892487`
`strand`	Strang	+ oder −
`frame`	Leserahmen (0, 1, 2)	`.`
`attributes`	optionale Angabe von Eigenschaften	`shift 132`
`comments`	optionale Angabe von Kommentaren zu dem Eintrag	

B.2 Sigma-Abtastung

Die Untersuchung wurde für die Genome von *E. coli*, *B. subtilis*, *P. aeruginosa*, *B. pseudomallei* und *R. solanacearum* mit einer Gättung im Bereich $\sigma \in \{0{,}1, 0{,}2, \ldots, 2{,}0\}$ durchgeführt. Die Performanz wurde jeweils im Vergleich zur Referenzannotation ermittelt (siehe 4.5.1, S. 64). Die jeweils beste Performanz ist hervorgehoben. Graphisch ist die Performanz in Abbildung 4.9, S. 71 dargestellt.

Tabelle B.2: **»Sigma-Abtastung«: Die Tabelle zeigt die Abhängigkeit der Vorhersageperformanz von TICO vom Gättungsparameter σ.**

σ	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0
EcoGene	93,2	93,2	93,4	93,8	**94,3**	94,2	94,2	93,4	93,4	93,6
B. subtilis non-y	89,1	89,2	**89,3**	**89,3**	**89,3**	**89,3**	89,2	88,9	88,5	88,6
PseudoCAP	79,6	79,6	80,2	82,5	84,8	**85,2**	84,9	85,0	84,7	84,6
B. pseudomomallei Chr. 1	66,4	66,4	66,8	68,9	69,5	**69,8**	69,7	69,5	69,1	68,4
B. pseudomallei Chr. 2	62,0	62,1	62,9	65,9	67,0	**67,2**	67,1	66,7	66,0	65,4
R. solanacearum Chr.	68,3	68,3	68,8	71,6	74,7	**75,2**	**75,2**	75,1	74,6	74,5
R. solanacearum Plasmid	60,7	60,7	61,8	65,4	70,1	**71,0**	70,1	69,7	69,5	68,8

σ	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9	2,0
EcoGene	93,2	93,0	92,5	92,6	92,3	91,3	91,5	89,3	87,5	86,0
B. subtilis non-y	88,6	88,8	88,8	88,6	88,5	88,5	88,1	87,0	84,9	79,2
PseudoCAP	84,4	84,1	83,6	83,2	82,9	82,5	80,3	73,4	57,9	56,8
B. pseudomallei Chr. 1	68,3	67,8	67,3	66,8	65,9	65,1	64,8	63,9	63,4	62,1
B. pseudomallei Chr. 2	65,6	65,4	64,8	64,6	64,3	63,9	63,8	61,3	58,1	55,2
R. solanacearum Chr.	74,0	73,8	73,4	73,3	72,8	72,2	71,6	70,9	70,0	69,2
R. solanacearum Plasmid	68,7	68,1	67,7	66,2	65,5	65,2	64,0	60,3	55,7	53,5

B.3 Verteilung der Startcodons

Tabelle B.3: **Prozentuale Startcodon-Usage der Gene, die von GLIMMER und TICO auf den untersuchten Genomen vorhergesagt wurde, im Vergleich zu den verwendeten Referenzdatensätzen.**

Organismus	*ATG*	*GTG*	*TTG*	*CTG*	andere
GLIMMER					
EcoGene	70,7	15,0	14,3	-	-
B.subtilis non-y	62,0	18,3	19,7	-	-
PseudoCAP	66,3	20,4	13,2	-	-
B. pseudomallei Chr. 1	33,2	29,5	22,2	-	-
B. pseudomallei Chr. 2	47,9	30,4	21,6	-	-
R. solanacearum Chr.	50,0	28,2	21,9	-	-
R. solanacearum Plasmid	48,6	28,7	24,6	-	-
TICO					
EcoGene	89,8	8,0	2,3	-	-
B.subtilis non-y	77,9	9,6	12,8	-	-
PseudoCAP	88,0	8,7	3,3	-	-
B. pseudomallei Chr. 1	64,1	18,7	17,2	-	-
B. pseudomallei Chr. 2	63,1	19,7	17,2	-	-
R. solanacearum Chr.	70,0	14,8	15,2	-	-
R. solanacearum Plasmid	62,3	18,3	19,3	-	-
Referenz					
EcoGene	90,6	7,4	1,8	-	0,2
B.subtilis non-y	79,7	9,0	11,1	0,1	0,1
PseudoCAP	87,1	9,1	1,4	-	2,4
B. pseudomallei Chr. 1	80,9	11,6	7,4	0,0	0,1
B. pseudomallei Chr. 2	81,2	12,2	5,8	0,1	0,1
R. solanacearum Chr.	86,1	9,2	4,5	0,0	0,1
R. solanacearum Plasmid	81,7	12,4	5,9	-	0,1

B.4 Verteilung der PWM-Scores

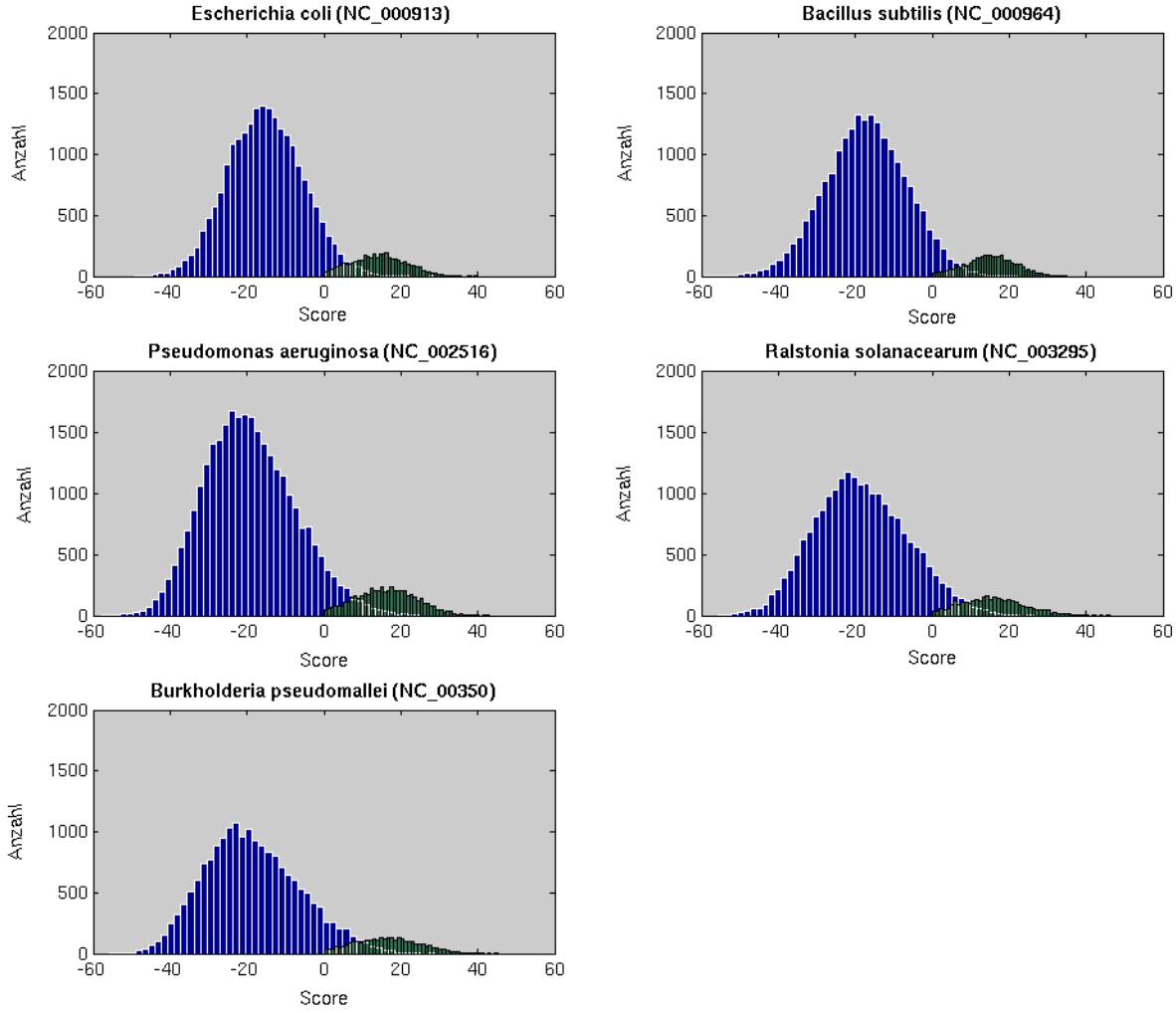

Abbildung B.1: **Histogramm der finalen Score-Verteilungen für die TIS-Kandidaten der in dieser Arbeit untersuchten Organismen. Die Scores aller Kandidaten der Kategorie *weak* sind blau dargestellt, die Scores von Kandidaten der Kategorie *strong* grün. Beide Kategorien sind durch jeweils 50 Histogrammbalken dargestellt. Diese Darstellung der Score-Verteilungen zeigt, dass sich die Kategorien anhand der Scores gut trennen lassen.**

Abbildungsverzeichnis

Literaturverzeichnis

[1] P. Meinicke, M. Tech, B. Morgenstern, and R. Merkl. Oligo kernels for datamining on biological sequences: A case study on prokaryotic translation initiation sites. *BMC Bioinformatics*, 5(169), 2004. http://www.biomedcentral.com/1471-2105/5/169.

[2] M. Tech and P. Meinicke. An unsupervised classification scheme for improving predictions of prokaryotic TIS. *BMC Bioinformatics*, 7(121), 2006. http://www.biomedcentral.com/1471-2105/7/121.

[3] M. Tech, N. Pfeifer, B. Morgenstern, and P. Meinicke. TICO: A tool for improving predictions of prokaryotic translation initiation sites. *Bioinformatics*, 17(21):3568–3569, 2005.

[4] M. Tech, B. Morgenstern, and P. Meinicke. TICO: A tool for postprocessing the predictions of prokaryotic translation initiation sites. *Nucleic Acids Res.*, 34:588 – 590, 2006.

[5] P.M. Sharp and W.-H. Li. An evolutionary perspective on synonymous codon usage in unicellular organisms. *J. Mol. Evol*, 24:28–38, 1986.

[6] R. Merkl. SIGI: score-based identification of genomic islands. *BMC Bioinformatics*, 5(22), 2004. http://www.biomedcentral.com/1471-2105/5/22.

[7] S. Waack, O.Keller, R.Asper, T.Brodag, C. Damm, W.F. Fricke, P. Meinicke K. Surovcik, and R. Merkl. Score-based prediction of genomic islands in prokaryotic genomes using hidden Markov models. *BMC Bioinformatics*, 7(142), 2006. http://www.biomedcentral.com/1471-2105/7/142.

[8] P. F. Hallin and D. W. Ussery. CBS Genome Atlas Database: a dynamic storage for bioinformatic results and sequence data. *Bioinformatics*, 20(18):3683–3686, 2004.

[9] W. Gish and D.J. States. Identifikation of protein coding regions by database similarity search. *Nat. Genet.*, 3:266–272, 1993.

[10] A. L. Delcher, D. Harmon, S. Kasif, O. White, and S. L. Salzberg. Improved microbial gene identification with GLIMMER. *Nucleic Acids Res*, 27(23):4636–4641, 1999.

[11] M. Borodovsky and D. Mcininch. GeneMark: Parallel Gene Recognition for both DNA Strands. *Comp. Chem.*, 17:123–133, 1993.

[12] H.-Y.Ou, F.-B. Guo, and C.-T. Zhang. GS-Finder: a program to find bacterial gene start sites with a self-training method. *The International Journal of Biochemistry and Cell Biology*, 36(3):535–544, 2004.

[13] H.-Q. Zhu, G.-Q. Hu, Z.-Q. Ouyang, J. Wang, and Z.-S. She. Accuracy improvement for identifying translation initiation sites in microbial genomes. *Bioinformatics*, 20(18):3308–3317, 2004.

[14] M. Tech and R. Merkl. YACOP: Enhanced gene prediction obtained by a combination of existing methods. *In Silico Biology*, 3(4):441–51, 2003.

[15] H. Nielsen, J. Engelbrecht, S. Brunak, and G. Heijne. Identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites. *Protein Engineering*, 10:1–6, 1997.

[16] Varshavsky A. The N-end rule: Functions, mysteries, uses. *Proc. Natl. Acad. Sci. USA*, 93:12142–12149, 1996.

[17] D. Barrik, K. Villanueba, J. Childs, R. Kalil, and T.D. Schneider. Quantitative analysis of ribosome binding sites in *E. coli*. *Nucleic Acids Res.*, 22:1287–1295, 1994.

[18] GenBank am National Center for Biotechnology Information (NCBI). http://www.ncbi.nlm.nih.gov/Genbank/index.html.

[19] SwissProt. http://www.expasy.org/sprot/.

[20] K. E. Rudd. EcoGene: a genome sequence database for *Escherichia coli* K-12. *Nucleic Acids Res.*, 28:60–64, 2000.

[21] F. R. Blattner, G. III Plunkett, C. A. Bloch, N. T. Perna, V. Burland, M. Riley, J. Collodo-Vides, D. D. Glasner, C. K. Rode, G. F. Mayhew, J. Gregor, N. W.Davis, H. A. Kirkpatrick, M. A. Goeden, D. J. Rose, B. Mau, and Ying Shao. Complete genome sequence of *Escherichia coli* K-12. *Science*, 277(5331):1453–1474, 1997.

[22] F. Kunst, N. Ogasawara, L. Moszer, A. M. Albertini, G. Alloni, V. Azevedo, M. G. Bertero, P. Bessieres, A. Bolotin, and S. BorchertR. Borriss *et. al*. The complete genome

sequence of the Gram-positive bacterium *Bacillus subtilis*. *Nature*, 390(6657):249–256, 1997.

[23] K.C. Stover, X.Q. Pham, A.L. Erwin, S.D. Mizoguchi, P. Warrener, M.J. Hickey, F.S.L. Brinkman, W. O. Hufnagle, D.J. Kowalik, and M. Lagrou. Complete genome sequence of *Pseudomonas aeruginosa* PAO1: an opportunistic pathogen. *Nature*, (406):959–964, 2000.

[24] PseudoCAP *Pseudomonas aeruginosa* Community Annotation Project. http://pseudomonas.com/ und http://v2.pseudomonas.com/.

[25] D.A. Berry. *Statistics – A Bayesian Perspective*. Duxbury Press, Belmont, California, 1996.

[26] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[27] N. Christiani and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods.* Cambridge University Press, 2000.

[28] R. Durbin, S. R. Eddy, and A. Krogh. *Biological Sequence Analysis*. Cambridge University Press, 1998.

[29] P. Baldi and S. Brunak. *Bioinformatics - The machine learning approach*. Massachusetts Institute of Technology Press, 1998.

[30] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statisical Learning*. Springer Verlag, 2001.

[31] R. Rifkin, G. Yeo, and T. Poggio. *Regularized Least Squares Classification. In Advances in Learning Theory: Methods, Model and Applications NATO Science Series III: Computer and Systems Sciences*, volume 190. IOS Press, Amsterdam, 2003.

[32] S. Lloyd. Least squares quantization in pcm. *Technical Report Bell Laboratories (1957), published IEEE Transactions on information theory*, 28(2):128–137, 1982.

[33] J. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifths Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

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

[35] A. G. Pedersen and H. Nielsen. Neural network prediction of translation initiation sites in eukaryotes: Perspectives for est and genome analysis. In *5th International Conference on Intelligent Systems for Molecular Biology*, pages 226–233, 1997.

[36] A. Zien, G. Rätsch, S. Mika, B. Schölkopf, T. Lengauer, and K.R. Müller. Engineering Support Vector Machine kernels that recognize translation initiation sites. *Bioinformatics*, 16(9):799–807, 2000.

[37] H. Li and T. Jiang. A Class of Edit Kernels for SVMs to Predict Translation Initiation Sites in Eukaryotic mRNAs. *RECOMB*, 2004.

[38] S. Degroeve, B. De Beats, Y. Van de Peer, and P. Rouzé. Feature subset selection for splice site prediction. *Bioinformatics*, 18 Suppl 2:75–83, 2002.

[39] C. Leslie, E. Eskin, and W.S. Noble. The Spectrum Kernel: A string kernel for SVM protein classification. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 564–575. Stanford, 2002.

[40] F. Markowetz, L. Edler, and M. Vingron. Support Vector Machines for protein fold class prediction. *Biometrical Journal*, 45(3):377–389, 2003.

[41] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[42] T. Joachims. Making large-scale support vector machine learning practical. In *Advances in Kernel Methods: Support Vector Machines*, pages 169–184. MIT Press, Cambridge, MA, 1998.

[43] A. J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning.

[44] S.L. Salzberg. A method for identifying splice sites and translationai start sites in eukaryotic mRNA. *As Computer Applications in the Biosciences (CABIOS)*, 13(4):365–376, 1997.

[45] J. Shine and L. Dalgarno. The 3' terminal sequence of *Escherichia coli* 16S ribosomal RNA: complementary to nonsense triplets and ribosom binding sites. *Proc. Natl. Acad. Sci.*, 71:1342–1346, 1974.

[46] J. Ma, A. Campbell, and S. Karlin. Correlation between Shine-Dalgarno sequences and gene features such as predicted expression levels and operon structures. *J. of Bacteriology*, 184(20):5733–5745, 2002.

[47] R. K. Shultzaberger, R. E. Buchheimer, K. E. Rudd, and T. D. Schneider. Anatomy of *Escherichia coli* ribosome binding sites. *J. Mol. Biol.*, 313:215–228, 2001.

[48] C. M. Stenstrom and L. A. Isaksson. Influences on translation initiation and early elongation by the messenger RNA region flanking the initiation codon at the 3' side. *Gene*, 288(1–2):1–8, 2002.

[49] C. M. Stenstrom, H. Jin, L. L. Major, W. P. Tate, and L. A. Isaksson. Codon bias at the 3'-side of the initiation codon is correlated with translation initiation efficiency in *Escherichia coli*. *Gene*, 263(1–2):273–284, 2001.

[50] T. Sato, M. Terabe, H. Watanabe, T. Gojobori, C. Hori-Takemoto, and K. Miura. Codon and base biases after the initiation codon of the open reading frames in the *Escherichia coli* genome and their influence on the translation efficiency. *J. Biochem.*, 129(6):851–60, 2001.

[51] I. Lebars, R. Hu, J. Lallemand, M. Uzan, and F. Bontems. Role of the substrate conformation and of the S1 protein in the cleavage efficiency of the T4 endoribonuclease RegB. *J Biol Chem*, 16(276):132647, 2001.

[52] D. E. Draper. Escherichia coli *and* Salmonella, volume I: Translational Initiation, pages 902–908. ASM Press, 2nd edition, 1996.

[53] T.D. Schneider and R.M. Stephens. Sequence Logos: A New Way to Display Consensus Sequences. *Nucleic Acids Res*, 18:6097–6100, 1990.

[54] G.E. Crooks, G. Hon, J.M. Chandonia, and SE Brenner. WebLogo: A sequence logo generator. *Genome Research*, 14:1188–1190, 2004. http://weblogo.berkeley.edu/.

[55] Leila Taher, Burkhard Morgenstern, and Peter Meinicke. Splice Site Prediction using SVM with the Oligo Kernel. In *12th Annual International Conference on Intelligent Systems for Molecular Biology (ISMB)*, 2004.

[56] B. Mersch, K.-H. Glatting, S. Suhai, and A. Hotz-Wagenblatt. Detecting exonic splice enhancers using machine learning techniques. In *15th Annual International Conference on Intelligent Systems for Molecular Biology (ISMB)*, 2007.

[57] C. Igel, T. Glasmachers, B. Mersch, N. Pfeifer, and P. Meinicke. Gradient-based Optimization of Kernel-Target Alignment for Sequence Kernels Applied to Bacterial Gene Start Detection. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4:216–226.

[58] N. Christiani, J. S. Shawe-Taylor, A. Elisseeff, and J. Kandola. On Kernel-Target Alignment. *Advances in Neural Information Processing Systems*, 14, 2001.

[59] B. Schölkopf, K. Tsuda, and J.-P. Vert. Kernel Methods in Computational Biology. 2004.

[60] M. Tompa. An exact method for finding short motifes in sequences, with application to the ribosom binding site problem. *Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 262–271, 1999.

[61] S. S. Hannenhalli, W. S. Hayes, A. G. Hatzigeoriou, and J. W. Fickett. Bacterial start site prediction. *Nucleic Acids Res.*, 27(17):3577–3582, 1999.

[62] B. E. Suzek, M. D. Ermolaeva, M. Schreiber, and S. L. Salzberg. A probabilistic method for identifying start codons in bacterial genomes. *Bioinformatics*, 17(12):1123–1130, Dec 2001.

[63] D. Frishman, A. Mironov, and M. Gelfand. Starts of bacterial genes: estimating the reliability of computational predictions. *Gene*, 234:257–265, 1999.

[64] P. Londei. Evolution of translational initiation: new insights from the archaea. *FEMS - Microbiology Reviews*, 29:185–200, 2004.

[65] A. Utsugi. Density estimation by mixture models with smoothing priors. *Neural Computation*, 10(8):2115–2135, 1998.

[66] Apache TOMCAT. http://tomcat.apache.org/. *Apache Software Foundation*.

[67] Java Servlet. http://java.sun.com/products/servlet/. *Sun Microsystems*.

[68] Java Server Pages (JSP). http://java.sun.com/products/jsp/. *Sun Microsystems*.

[69] Trust Sanger Institute. http://www.sanger.ac.uk/.

[70] K. Rutherford, J. Parkhill, J. Crook, T. Horsnell, P. Rice, M-A. Rajandream, and B. Barrell. Artemis: sequence visualisation and annotation. *Bioinformatics*, 16(10):944–945, 2000.

[71] T. Yada, Y. Totoki, T. Takagi, and K. Nakai. A novel bacterial gene-finding system with improved accuracy in locating start codon. *DNA Res.*, 8:97–106, 2001.

[72] T.J. Carver, K.M. Rutherford, M. Berriman, M.-A. Rajandream, BG Barrell, and J.Parkhill. ACT: the Artemis comparison tool. *Bioinformatics*, 21(16):3422–3423, 2005.

[73] T. Schiex, J. Gouzy, A. Moisan, and Y. de Oliveira. FrameD: a flexible program for quality check and gene prediction in prokaryotic genomes and noisy matured eukaryotic sequences. *Nucl. Acids. Res.*, 31:3738–3741, 2003.

[74] J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. 2006.

[75] R. Zhang and C.T. Zhang. Z curves, an intuitive tool for visualizing and analyzing the DNA sequences. *Biomolecular Structure & Dynamics*, 11:767–783, 2004.

[76] Göttingen Genomics Laboratory (G2L). http://www.g2l.bio.uni-goettingen.de/.

[77] E. E. Snyder, N. Kampanya, J. Lu, E. K. Nordberg, H. R. Karur, M. Shukla, J. Soneja, Y. Tian, T. Xue, H. Yoo, and F. Zhang *et. al*. PATRIC: The VBI PathoSystems Resource Integration Center. *Nucleic Acids Res.*, 35:D401 – D406, 2007.

[78] Boyu Yang and Tian Xue and Jing Zhao and Chaitanya Kommidi and Jeetendra Soneja and Jian Li and Rebecca Will and Bruce Sharp and Ron Kenyon and Oswald Crasta and Bruno W. S. Sobral. Bioinformatics Web Services. 2006. http://pathport.vbi.vt.edu/services/wsdls/tico.wsdl.

Lebenslauf

Name	Maike
Nachname	Tech
Geboren	03.09.1977

Schulausbildung

1984–1988	Besuch der Gerhard-Hauptmann-Schule Stockelsdorf
1988–1997	Besuch des Leibniz-Gymnasiums Bad Schwartau
1997	Abitur, Note ›gut‹

Studium

1997–2003	Studium der Biologie an der Georg-August-Universität Göttingen
1999	Vordiplom, Note ›sehr gut‹
2003	Diplom, Note ›sehr gut‹
Hauptfach	Mikrobiologie
Nebenfächer	Immunologie, Informatik
Thema der Diplomarbeit	»Vergleichende Untersuchung zur Verbesserung von Methoden der Genvorhersage«
2003–2007	Promotion

Berufliche Laufbahn

1997/1998	Teilzeittätigkeit im PKW-Check-In bei der Securitas GmbH Sicherheitsdienste
2000–2003	Teilzeittätigkeit als Anwendungsentwicklerin für die Prof. Schumann GmbH
seit 2003	Wissenschaftliche Mitarbeiterin der Abteilung Bioinformatik am Institut für Mikrobiologie und Genetik der Georg-August-Universität Göttingen