Technische Fakultät der Universität Bielefeld, AG Neuroinformatik

Neue Ansätze zum maschinellen Lernen von Alignments

Diplomarbeit

vorgelegt von

Ulf Großekathöfer
geboren am 15.04.1974
Matrikelnummer: 1375770
Universität Bielefeld

Thomas Lingner
geboren am 30.04.1977
Matrikelnummer: 1404876
Universität Bielefeld

---

betreut von

Prof. Helge Ritter
Dr. Peter Meinicke

Bielefeld, 28. September 2005

Diese Arbeit beinhaltet folgende Aufteilung der Einzelleistungen:

Thomas Lingner: Seiten 7-9, 11-12, 17-18, 21-26, 31-32, 37-43, 47-60, 73-76, 81-82, 87-96, 103-105, 109-111, 114-115, 123-124,

Ulf Großekathöfer: Seiten 9-10, 13-16, 19-20, 27-30, 33-36, 44-46, 61-72, 77-80, 83-86, 97-102, 106-108, 112-113, 116-122, 125-131.

Inhaltsverzeichnis

Zusammenfassung

Standardmethoden des maschinellen Lernens gehen davon aus, dass die benötigten Lernbeispiele bereits als Elemente in einem einheitlichen Vektorraum vorliegen. Die Einbettung von Zeitserien und Sequenzen in einen solchen Vektorraum stellt sich jedoch im Allgemeinen aufgrund von unterschiedlichen Längen und variierender zeitlicher Entwicklung als ein schwieriges Problem dar.

Bisherige Ansätze zur Vektorisierung bzw. nichtlinearen Einbettung der Daten neigen dazu, Informationen zu eliminieren, Korrelationen zu vernachlässigen, schwer interpretierbar oder rechenaufwändig zu sein.

In dieser Arbeit stellen wir zwei neue Ansätze zum Lernen auf der Grundlage von „rohen“ nicht-vektorisierten Zeitserien und Sequenzen vor. Während die *Ordered-Means-Modelle* auf generativen endlichen Zustandsautomaten basieren, die über die Maximierung der Likelihood optimiert werden, realisieren die *Feature-Alignment-Maschinen* eine adaptive Merkmalsselektion, die sich in bestehende Verfahren des maschinellen Lernens integrieren lässt.

Unsere Ergebnisse auf den Proteinsequenzen und den EEG-Daten sind vielversprechend und zeigen die Eignung der vorgestellten Ansätze zur domänenübergreifenden Sequenzanalyse.

1. Einleitung

1.1. Motivation

In vielen Anwendungsgebieten haben sich in den letzten Jahrzehnten Methoden des maschinellen Lernens als geeignete Analysewerkzeuge etabliert. Eine Bedingung der meisten Verfahren ist, dass die zu untersuchenden Daten in einem einheitlichen Vektorraum liegen müssen, in dem jede Dimension mit einem bestimmten Merkmal identifiziert ist. Jedoch ist eine solche Abbildung der Messwerte (bzw. Dateneinträge) auf Merkmale in den meisten Fällen nicht gegeben. Beispiele dafür sind Zeitserien (z.B. natürlichsprachliche Äußerungen, Bewegungstrajektorien) und Sequenzen (z.B. Aminosäureketten), für die keine allgemeingültige Vorschrift zur Einbettung in den erforderlichen Vektorraum existiert.

Liegen die Daten nicht im selben Merkmalsraum oder kommt es auf einen (möglicherweise variierenden) zeitlichen Verlauf an, so ist eine nichtlineare zeitliche Transformation (Alignment, Vektorisierung) erforderlich, um die Daten verarbeiten zu können. Die Standardverfahren des maschinellen Lernens zur Vektorisierung berücksichtigen dabei oft die variierenden Positionsinformationen und eventuelle Korrelationen nur ungenügend, was sich z.B. in reduzierter Analyseleistung niederschlägt.

Mit dem folgenden Beispiel wird das Problem deutlich. Die in Abbildung 1.1 notierten Proteinsequenzen stammen aus dem von uns untersuchten SCOPSUPER95_66-Datensatz. Sie variieren stark in der Länge ihrer Aminosäureketten.

```
>d1f5wa_ b.1.1.1 (A:) Coxsackie virus and adenovirus receptor (Car), domain 1 {Human (Homo sapiens)}

farslsittpeemiekakgetaylpckftlspedqgpldiewlispadnqkvdqviilysgdkiyddyypdlkgrvhftsndlksgdasinvtnlqlsdigtyqc
kvkkapgvankkihlvvlv

>d1wioa4 b.1.1.3 (A:292-363) CD4 {Human (Homo sapiens)}

mratqlqknltcevwgptspklmlslklenkeakvskrekavwvlnpeagmwqcllsdsgqvllesnikvlp

>d1f2qa2 b.1.1.4 (A:86-174) IgE high affinity receptor alpha subunit {Human (Homo sapiens)}

dwlllqasaevvmegqplflrchgwrnwdvykviyykdgealkywyenhnisitnatvedsgtyyctgkvwqldyeseplnitvikapr
```

Abbildung 1.1.: Drei Proteinsequenzen des von uns verwendeten SCOPSUPER95_66-Datensatzes (s. Abschnitt 7.1) mit Beschreibung im FASTA-Format.

Eine Möglichkeit zur Vereinheitlichung besteht darin, die Länge des Merkmalsvektors für alle Sequenzen auf die Länge der kürzesten Sequenz (hier Sequenz 2) festzulegen und damit längere Sequenzen abzuschneiden. Dabei gehen jedoch Informationen

unwiederbringlich verloren. Vor allem aber werden damit korrespondierende Merkmale (Aminosäuren) an unterschiedlichen Positionen außer Betracht gelassen.

Eine andere Herangehensweise ist, Sequenzmerkmale (z.B. Oligomere, d.h. Folgen einzelner Elemente) zu zählen und entsprechend der Kombinationsmöglichkeiten in einem Vektor zusammenzustellen (s. Abb. 1.2). Diese Methode ist jedoch aufgrund der mit der Anzahl der Werte (bzw. Symbole) exponentiell wachsenden Anzahl der Kombinationen nur für wenige Anwendungen mit kleinem Wertebereich geeignet. Vor allem aber werden sämtliche Positionsinformationen verworfen.

$$\begin{matrix} n_{far} = \\ n_{ars} = \\ n_{rsl} = \\ \vdots \end{matrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \end{bmatrix}$$

Abbildung 1.2.: Ausschnitt der Oligomerhäufigkeitsrepräsentation der ersten Sequenz aus Abbildung 1.1 für Trimere (Kombination von drei Elementen).

Eine weitere Möglichkeit besteht darin, alle Sequenzen auf die Länge der längsten Sequenz zu strecken, indem die kürzeren Sequenzen durch Lücken aufgefüllt werden. Allerdings ist nicht klar, wo die Lücken platziert werden müssen und wie sie im Vektorraum sinnvoll repräsentiert werden können. Diese Methode entspricht einem naiven *multiplen Alignment*. Ausgereifte Varianten berücksichtigen Beziehungen zwischen den Elementen (z.B. Austauschwahrscheinlichkeiten von Aminosäuren), um abschnittsweise Ähnlichkeiten von Sequenzen mit einzubeziehen. Ein multiples Alignment für das Beispiel ist in Abbildung 1.3 zu sehen.

```
d1f5wa_         FARSLSITTPEEMIEKAKGETAYLPCKFTLSPEDQGPLDIEWLISPADNQKVDQVIILYS
d1f2qa2         --DWLLLQASAEVVME--GQPLFLRC--------HG-----W-----RNWDVYKVIYYKD
d1wioa4         -------MRATQLQKN-------LTCEV-------------WG---PTSPKLMLSLKLEN
                         . ::  :       * *               *      . .:   :   .

d1f5wa_         GDKIYDDYYPDLKGRVHFTSNDLKSGDASINVTNLQLSDIGTYQCKVKK-APGVANKKIH
d1f2qa2         GEAL--KYW--------YENHNIS-------ITNATVEDSGTYYCTGKVWQLDYESEPLN
d1wioa4         KEAK-------------VSKREKA--------VWVLNPEAGMWQCLLSD--SGQVLLESN
                 :                 ..:          .     : * : *  .    .      :

d1f5wa_         LVVLV---
d1f2qa2         ITVIKAPR
d1wioa4         IKVLP---
                : *:
```

Abbildung 1.3.: Mit ClustalW (s. Abschnitt 3.3) erstelltes multiples Alignment der drei Beispielsequenzen. Unter den Sequenzen ist die Konsensuszeile zu sehen. Wenig konservierte Positionen sind durch Punkte, mittelere durch Doppelpunkte und stark konservierte durch Sterne gekennzeichnet.

1.2. Bisherige Ansätze

Für Zeitserien (insbesondere Sprachsignale) hat sich zuerst das *Dynamic Time Warping* (DTW, s. [SC78]) als geeignet erwiesen. Dabei wird eine Zuordnungsfunktion durch ein dynamisches Programm berechnet (s. Abschnitt 3.1). Für Sequenzen wie in Abbildung 1.1 bieten sich *multiple Sequenz-Alignments* (MSA) als Mittel an, um die Sequenzen durch Einfügen von Lücken auf dieselbe Länge zu bringen (s. Abb. 1.3 und Abschnitt 3.3). Beide Verfahren eignen sich jedoch ausschließlich als eigenständige Vorverarbeitungsschritte und sind keine Lernverfahren.

Nachfolger des DTW-Algorithmus sind die *Hidden-Markov-Modelle* (HMM, s. [Rab89], Abschnitt 2.5), welche ein iteratives Lernschema enthalten und in der Lage sind, Zeitserien zu segmentieren und zu analysieren. Für Proteinsequenzen wurden die HMMs erweitert und die Familie der *Profile-Hidden-Markov-Modelle* (PHMMs, s. [Kro93], [Kro94], Abschnitt 3.4) entwickelt. HMMs sind durch ihre zahlreichen Modellvarianten und Parameter schwierig zu konfigurieren und neigen (besonders bei wenigen Trainingsdaten) zur Überparametrisierung. Es besteht keine direkte Möglichkeit, multiple Alignments aus HMMs zu gewinnen, was sie in diesem Sinne schwer interpretierbar macht. Auch zeitliche Korrelationen in Zeitserien sind nicht mit HMMs modellierbar. Viele PHMMs setzen zudem eine Initialisierung durch ein bestehendes multiples Alignment voraus.

Im Zuge des Erfolgs der kernbasierten Lernverfahren wurden in letzter Zeit sogenannte *Sequenzkerne* (s. [Jaa98], [Lai02], [Tsu02], [Gor03], [Sai04]) erfolgreich eingesetzt. Diese erzeugen einen abstrakten erweiterten Merkmalsraum und bewerten die Ähnlichkeit zweier Daten mittels sogenannter Kernfunktionen. Im erweiterten Merkmalsraum (*Feature Space*) ist die Analyse von Korrelationen möglich. Schwachpunkt dieser Verfahren ist die Notwendigkeit, jedes Beispiel mit jedem anderen Beispiel unter Benutzung der Kernfunktion auszuwerten. Aufgrund dieser quadratischen Komplexität in Abhängigkeit von der Anzahl der verwendeten Daten eignen sich Kernmethoden (s. Abschnitt 3.5) nur bedingt für große Datenmengen.

Weitere Ansätze zur Vektorisierung von Sequenzen (s. [Mar03], [Les04]) verwerfen jegliche Positionsinformation durch die Reduktion auf das Auszählen der Auftrittshäufigkeit einzelner Sequenzmerkmale. Als Konsequenz sind sie prinzipbedingt im Nachteil, wenn Positionsabhängigkeiten eine wichtige Informationsquelle sind.

1.3. Neue Ansätze

In dieser Diplomarbeit stellen wir zwei Ansätze vor, die maschinelles Lernen auf der Grundlage von Sequenzen schneller und genauer realisieren als etablierte Verfahren und dabei sowohl die variierenden Positionsinformationen als auch zeitliche Korrelationen berücksichtigen. Dabei ist das Alignment kein isolierter Vorverarbeitungsschritt, sondern Bestandteil der Lernverfahren. Dadurch erwarten wir, dass die Methoden bessere Ergebnisse als bisherige Ansätze erzielen, was z.B. signifikant höhere Generalisierungsleistungen bei der Sequenzklassifikation bedeuten würde.

Da die Verfahren modellbasiert lernen, erwarten wir diese Modelle intuitiv interpretie-

ren zu können und dadurch weitere Erkenntnisse über die zugrundeliegenden Sequenzen und Sequenzklassen zu ermöglichen.

Besonderen Wert legen wir auf die Domänenunabhängigkeit der Algorithmen. Die von uns entwickelten Verfahren sollen auf unterschiedliche Sequenzarten anwendbar sein und auch in Zukunft neue Domänen erschließen können.

1.4. Aufbau der Arbeit

In den beiden nachfolgenden Kapiteln beschreiben wir die theoretischen Grundlagen des maschinellen Lernens (Kapitel 2) und des Alignments (Kapitel 3). Nachdem wir in Kapitel 4 kurz und überblickhaft die beiden neuen Ansätze gegeneinander und gegenüber anderen verwandten Methoden abgrenzen, beschreiben wir deren Theorie ausführlich in Kapitel 5 (Ordered-Means-Modelle) sowie Kapitel 6 (Feature-Alignment-Maschinen). Kapitel 7 beschreibt die von uns verwendeten Datensätze und durchgeführten Experimente. Die Ergebnisse, die in Kapitel 8 zusammengefasst sind, werden in Kapitel 9 diskutiert, worauf Kapitel 10 dann das Fazit bildet. Im Anhang gehen wir auf Implementationsdetails (A) und die Notation (B) ein.

2. Methoden des maschinellen Lernens

2.1. Einführung in das maschinelle Lernen

Was ist maschinelles Lernen?

Lernen ist die auf Erfahrung basierende Veränderung eines Systems dahingehend, dass es ähnliche Aufgaben nach diesem Prozess besser oder schneller bewältigen kann (s. [Nil96], [Lan96]). Nach [CM98] ist maschinelles Lernen die „Schätzung unbekannter Abhängigkeiten oder Strukturen eines Systems durch begrenzte Anzahl von Beobachtungen". Ziel des maschinellen Lernens (ML) ist es, Aspekte menschlicher Lernfähigkeit auf Algorithmen und Computerprogramme (im Allgemeinen auf Maschinen) zu übertragen. Zur Bewertung des Erfolgs gibt es verschiedene anwendungs- und methodenabhängige Qualitätsmaße (*performance measures*), auf die wir später noch näher eingehen werden.

Die verschiedenen Arten des Lernens entstehen aus den Varianten von Modellstrukturen, also dem, *was* gelernt wird, und den Möglichkeiten, wie die Erfahrungen – hier in Form von Trainingsbeispielen – präsentiert werden. Mögliche Modellstrukturen umfassen Funktionen, logische Programme, endliche Automaten, formale Grammatiken, allgemeine Problemlöser und vieles mehr. In unserer Arbeit steht das Lernen von Funktionen im Vordergrund. Man kann unterscheiden zwischen dem *überwachten Lernen* von Funktionen (*supervised learning*), bei welchem jedem Beispiel Sollausgaben zugeordnet sind, dem *unüberwachten* oder *unsupervised* Fall ohne Sollausgaben sowie dem *Verstärkungslernen* (*reinforcement learning*), bei dem ein qualitatives Feedback (interpretierbar als Belohnung oder Bestrafung) optimiert werden soll. Das Ziel ist in jedem Fall, die Ausgaben zukünftiger Beispiele anhand der Modelle möglichst genau vorherzusagen. Diese *Generalisierungsfähigkeit* kann z.B. mittels Kreuzvalidierung (s. Abschnitt 2.9) getestet werden. Wir beschränken uns im Folgenden auf das unüberwachte bzw. überwachte Lernen von Funktionen.

Das Gebiet des maschinellen Lernens tangiert viele andere Bereiche wie z.B. Statistik, Datenbanken, Künstliche Intelligenz, Neuronale Netze, (adaptive) Kontrolltheorie, Informations- und Komplexitätstheorie, Psychologie (besonders die Kognitionswissenschaften), Evolutionsbiologie (genetische Algorithmen) und sogar die Philosophie.

Ähnlich umfassend stellen sich die Anwendungsmöglichkeiten dar, die vom Einsatz in der Medizin (z.B. zur Diagnose), Sprach- und Objekterkennung und Wirtschaftswissenschaften (Marketing) bis hin zur Biologie (speziell Genetik) reichen.

Die wohl bekannteste Anwendung des ML ist das *Data Mining*, die Analyse von Daten zwecks Entdeckung neuen Wissens. Beim Data Mining werden große, teilweise unstrukturierte und hochdimensionale Datenmengen auf Regularitäten und Muster untersucht,

um aussagekräftige, interpretierbare und wertvolle Informationen zu gewinnen. Maschinelles Lernen stellt eine Basis von Algorithmen für das Data Mining bereit.

Als Teilgebiete des maschinellen Lernens zählen

- Regression (das Lernen und Vorhersagen von Funktionen),
- Klassifikation (Spezialfall der Regression mit diskreten Ausgaben),
- Visualisierung,
- Clustering und Vektorquantisierung (Partitionieren von Datenräumen) und
- Dichteschätzung.

Dabei kommen u.a. Techniken wie Neuronale Netze, Bayes-Klassifikatoren, Entscheidungsbäume, Genetische Algorithmen und endliche Zustandsautomaten zum Einsatz.

2.2. Merkmalsextraktion und -selektion

Um die Beobachtungen, welche die Eingabe für eine Technik des ML darstellen, zu repräsentieren, müssen wir die Eigenschaften der Daten durch *Attribut-Wert-Paare* charakterisieren. Damit die so entstehenden *Merkmale* als Datenbasis für einen Algorithmus verwendet werden können, müssen sie aus den Beobachtungen gewonnen (Merkmalsextraktion) und nach Relevanz ausgesucht (Merkmalsselektion) werden.

Ziel ist eine Repräsentation der Beobachtungen $O = o_1, .., o_N$ in maschinenverwertbarer Form, z.B. als Menge von *Merkmalsvektoren* $X = \{\mathbf{x}_1, .., \mathbf{x}_N\}$ aus dem $\mathbb{R}^d$ (*Merkmalsraum*) mit d Attributen.

Außerdem dient dieser Vorverarbeitungsschritt der Erkennung und Entfernung von *Ausreißern*. Ausreißer sind solche Datenwerte, die nicht konsistent mit dem Großteil der meisten anderen beobachteten Daten sind. Gründe dafür können z.B. Messfehler oder abnormale Abweichungen sein. Können die Ausreißer in diesem Schritt nicht identifiziert werden, ist es wichtig, robuste Modellierungsmethoden zu verwenden, da die Modelle sonst verfälscht werden können.

2.2.1. Merkmalsextraktion

Um die Beobachtungen in oben genannte Form zu bringen, müssen diese in einen Merkmalsraum transformiert werden. Dies beinhaltet sowohl die Aufnahme (Digitalisierung und Quantisierung) als auch die Vorverarbeitung (Segmentierung und Skalierung) der Messdaten. Wichtigster Schritt jedoch ist die Zusammenstellung der Merkmalsvektoren nach anwendungsabhängigen Berechnungsvorschriften. Die Dimensionalität d der Merkmalsvektoren – also die Anzahl der Merkmale – wirkt sich auf die Leistung und den Rechenaufwand des eingesetzten Verfahrens aus.

2.2.2. Merkmalsselektion

Hierbei geht es um die Identifikation und Auswahl relevanter Attribute der Beobachtungen. Dazu gehört u.a. die anwendungsabhängige Reduktion auf gemeinsame (Clustering) bzw. unterschiedliche (Klassifikation) Merkmale und ggf. die Vervollständigung oder Entfernung unvollständiger Beobachtungsdaten. Wenn die Identifikation relevanter Attribute Ziel des eingesetzten Verfahrens ist, bzw. das benötigte Wissen über die Relevanz fehlt, kann dieser Schritt die Auswahl aller möglichen zur Verfügung stehenden Merkmale bedeuten.

2.3. Einführung in die Klassifikation

Als Klassifikation bezeichnet man im Allgemeinen ein Ordnungsprinzip, welches einen abgegrenzten Gegenstandsbereich in mehrere Klassen (Kategorien) einteilt. Während die Objekte einer Klasse die gleichen Merkmale aufweisen, unterscheiden sich die Objekte anderer Klassen von diesen durch mindestens ein Merkmal. Die Klassifikation kann hierarchisch (analytisch und eindimensional) oder ahierarchisch (facettenhaft und multidimensional), manuell oder automatisch, sowie überwacht oder unüberwacht stattfinden. Dabei wird nicht hierarchisch wie z.B. bei der Taxonomie von Organismen vorgegangen, sondern anhand eines mehrdimensionalen Merkmalsvektors entschieden.

Ein Merkmalsvektor $\mathbf{x} \in \mathbb{R}^d$ setzt sich aus mehreren einzelnen Merkmalen $\mathbf{x} = [x_1, .., x_d]^T$ zusammen. Merkmalsvektoren jeder (manuell zusammengestellten) Klasse werden mit einer Kategoriebezeichnung (*label*) y versehen und als Trainingsmenge zusammengefasst. Diese wird benutzt, um den Klassifikator zu trainieren.

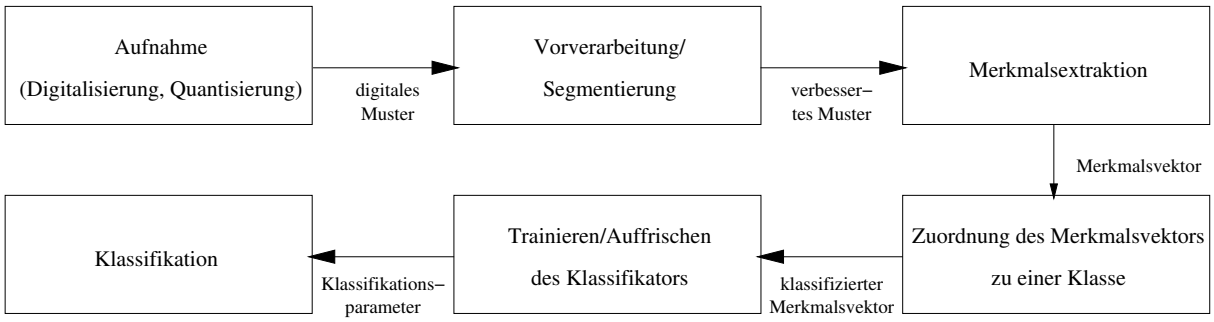

Abbildung 2.1.: Schematische Darstellung eines überwachten Klassifikationssystems.

Ein überwachtes Klassifikationssystem (siehe Abb. 2.1) besteht aus einer Vorverarbeitungsphase, in der die Daten aufgenommen und die Merkmale extrahiert werden, einer Trainingsphase, in welcher die Merkmalsvektoren dem Klassifikator zum Schätzen oder Berechnen der Klassifikationsparameter dienen und einer Test- bzw. Anwendungsphase, in welcher der trainerte Klassifikator zur Bestimmung der Kategorie unbekannter Merkmalsvektoren verwendet wird.

Ein Klassifikator f kann formal als eine Abbildung beschrieben werden, die einem Merkmalsvektor $\mathbf{x} \in \mathbb{R}^d$ eine von endlich vielen diskreten Klassen $y_i \in Y$ zuordnet:

$$f : \mathbb{R}^d \rightarrow Y. \tag{2.1}$$

Dabei bezeichnet Y mit $|Y| = M,\ M \in \mathbb{N}$ die diskrete Menge möglicher Klassifikationsergebnisse.

Bayesklassifikator

Ein klassischer Ansatz zur Lösung von Klassifikationsaufgaben ist der Bayesklassifikator, welcher das Prinzip der Risikominimierung verfolgt. Bei dieser wird versucht, das Risiko R einer Fehlklassifikation, ausgehend von einer Verlustfunktion L, in welcher die Kosten dafür anwendungsabhängig veranschlagt werden, so gering wie möglich zu halten. Formal gilt

$$\min R = E\{L\} = E\{L(f(\mathbf{x}), y_i)\}, \tag{2.2}$$

wobei E den Erwartungswert bezeichnet. Beim Bayesklassifikator kommt eine Verlustfunktion mit gleichen Kosten für eine Fehlklassifikation (und sonst 0) zum Einsatz.

Mit der klassenspezifischen Dichte $p(\mathbf{x}|y_i)$ und den a-priori-Wahrscheinlichkeiten $P(y_i)$ der Klassen $y_i \in Y$ kann, basierend auf dem Prinzip der Risikominimierung, gezeigt werden, dass

$$P(y_i|\mathbf{x}) = \frac{p(\mathbf{x}|y_i)P(y_i)}{p(\mathbf{x})} \tag{2.3}$$

gilt. Klassifiziert wird nach der größten a-posteriori-Wahrscheinlichkeit

$$f(\mathbf{x}) = \arg\max_{y_i} P(y_i|\mathbf{x}). \tag{2.4}$$

Da die gemeinsame Dichte $p(\mathbf{x})$ für alle Klassen gleich ist, reicht

$$f(\mathbf{x}) = \arg\max_{y_i} p(\mathbf{x}|y_i)P(y_i) \tag{2.5}$$

zur Lösung aus. Wenn die $p(\mathbf{x}|y_i)$ und die $P(y_i)$ unbekannt sind, müssen sie aus den gegebenen Daten geschätzt werden.

Eine gute Einführung in die Risikominimierung und Bayesklassifikation findet sich in [DHS01].

2.4. Prototypenklassifikation

Klassifikationsaufgaben lassen sich, wie in Abschnitt 2.3 motiviert, durch Dichteschätzung lösen. Aber es existieren auch alternative Lösungsprinzipien. Anstatt die gelabelte Stichprobe zur Schätzung von Dichten (Abschnitt 2.6) oder Trennfunktionen (Abschnitt 2.7) heranzuziehen, können die Elemente der Stichprobe direkt zur Klassifikation benutzt werden. Es wird kein Modell gelernt, das Verfahren ist speicherbasiert.

Nächster-Nachbar-Klassifikator

Der prominenteste Vertreter dieser Klassifikatoren ist der Nächster-Nachbar-Klassifikator (*nearest-neighbor classifier*). Ein ungelabelter Datenpunkt $\mathbf{x}$ wird der Klasse des nächstgelegenen, mit y_i gelabelten Trainingsbeispiels $\mathbf{x}_i$ zugewiesen:

$$f(\mathbf{x}) = y_i, \text{ falls } m(\mathbf{x}, \mathbf{x}_i) = \min_{n=1,..,N} m(\mathbf{x}, \mathbf{x}_n). \tag{2.6}$$

Dazu muss ein Distanzmaß $m(.)$ definiert werden. Verbreitet ist die euklidische Distanz

$$m(\mathbf{x}, \mathbf{z}) = \left(\sum_{l=1}^{d} (x_l - z_l)^r \right)^{\frac{1}{r}} \quad \text{mit } r = 2, \tag{2.7}$$

aber natürlich können hier auch andere Metriken ($r = 1, .., \infty$) verwendet werden.

Eine naheliegende Verbesserung des NN-Klassifikators ist es, die k nächsten Nachbarn zur Klassifikation heranzuziehen. Dadurch wird der Einfluss einzelner Ausreißer in den Trainingsdaten verringert.

Vektorquantisierung

Der NN-Klassifikator hat den Nachteil, dass zur Klassifikation die Distanz zu jedem gelabelten Element der Stichprobe berechnet werden muss. Dies kann unter Umständen sehr zeitintensiv werden. Daher ist es sinnvoll, die Lernstichprobe durch einzelne oder einige wenige Prototypen möglichst gut zu repräsentieren. Ein etabliertes Verfahren, das dies ermöglicht, ist die Vektorquantisierung.

Vektorquantisierer versuchen, den Datenraum in kompakte Gebiete aufzuteilen, die sich prototypisch repräsentieren lassen. Es haben sich eine Vielzahl von Verfahren zur Vektorquantisierung etabliert, darunter harte und weiche, hierarchische, divisive und agglomerative Vektorquantisierungsalgorithmen. Dazu kommen Optimierungsverfahren, die anhand eines geeigneten Maßes die Daten partitionieren (vgl. [CM98]).

Exemplarisch stellen wir hier den *k-means Algorithmus* vor. Das Ziel ist, jedes von N Datenelementen genau einem von $K < N$ Repräsentanten zuzuordnen und so den Datenraum zu partitionieren.

Eine mögliche Form der zu minimierenden Fehlerfunktion ist

$$E_{k-means} = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} h_{ij} ||\mathbf{x}_i - \boldsymbol{\mu}_j||^2 \tag{2.8}$$

$$\text{mit } h_{ij} = \begin{cases} 1, \text{ falls } j = y_i, \\ 0, \text{ sonst.} \end{cases} \tag{2.9}$$

Ein $h_{ij} \in 0, 1$ realisiert dabei die deterministische Zuordnung des i-ten Datenelements zum j-ten Prototypen. Zur Optimierung wird ein iteratives Projektions-Regressions-Schema verwendet. Der Algorithmus ist in Abbildung 2.2 skizziert.

1. Initialisierung der Prototypen. Initialisiere die Prototypen z.B. durch

$$\boldsymbol{\mu}_j \;=\; \mathbf{x}_j \text{ mit } j = 1, .., K$$

oder zufällig.

2. Optimierung der Zuordnungsvariable. Wähle die Zuordnungsvariable h_{ij} so, dass $E_{k-means}$ minimal wird:

$$h_{ij} \;=\; \begin{cases} 1, \text{ falls } j = \arg\min_J ||\mathbf{x}_i - \boldsymbol{\mu}_J||^2 \\ 0, \text{ sonst} \end{cases} \text{, für alle } i = 1, .., N.$$

3. Optimierung der Prototypen. Wähle die $\boldsymbol{\mu}_j$ so, dass $E_{k-means}$ minimal wird:

$$\begin{aligned} \nabla_{\boldsymbol{\mu}_j} E_{k-means} &= \frac{1}{N} \sum_{i=1}^{N} h_{ij}(2\boldsymbol{\mu}_j - 2\mathbf{x}_i) \stackrel{!}{=} \mathbf{0} \\ \Rightarrow \boldsymbol{\mu}_j &= \frac{\sum_{i=1}^{N} h_{ij}\mathbf{x}_i}{\sum_{i=1}^{N} h_{ij}}. \end{aligned}$$

4. Abbruch der Optimierung. Beende die Optimierung, wenn die Veränderung der Fehlerfunktion unterhalb eines Schwellwertes ϵ liegt:

$$\begin{array}{lll} \text{Falls} & |\Delta E_{k-means}| < \epsilon & \rightarrow \text{ Abbruch,} \\ \text{sonst} & & \rightarrow \text{ weiter bei 2.} \end{array}$$

Abbildung 2.2.: Iteratives Optimierungsschema des k-means Algorithmus.

Der k-means Algorithmus terminiert zwar sicher, aber es ist ungewiss, ob er auch das globale Minimum der Fehlerfunktion $E_{k-means}$ erreicht. Es ist ratsam, den Algorithmus mehrmals mit unterschiedlichen Initialisierungen zu starten. Neben *k-means* ist der Algorithmus unter verschiedenen anderen Namen bekannt, beispielsweise *LBG* oder *Generalized Lloyd*.

Mittelwertklassifikator

$K = 1$ überführt den Prototypen des k-means Algorithmus in den Mittelwertvektor, einen der einfachsten, denkbaren Prototypen. Durch Anwendung auf jede Klasse ergeben sich somit M Mittelwertvektoren. Das ungelabelte Testdatum wird derjenigen Klasse

zugeordnet, deren Mittelwertvektor dem Testdatum am nächsten liegt:

$$f(\mathbf{x}) = y_j, \text{ falls } ||\mathbf{x} - \boldsymbol{\mu}_j||^2 = \min_{J=1,..,M} ||\mathbf{x} - \boldsymbol{\mu}_J||^2. \tag{2.10}$$

2.5. Hidden-Markov-Modelle

Hidden-Markov-Modelle (HMM) sind generative probabilistische Modelle und können als generierender endlicher Automat mit beobachtbaren Emissionen aus versteckten (*hidden*) Zuständen aufgefasst werden. Sie haben sich bei der Signalanalyse von Sequenzen variabler Länge bewährt, da sie Segmentierungsinformationen und ein (wahrscheinliches) Modell der Daten aufgrund von Beobachtungsfolgen liefern.

Ein stochastischer Prozess, welcher als Folge diskreter Zustände $\mathbf{q} = q_1..q_T$ aus einer endlichen Menge von Zuständen $q_i \in \{1, .., K\}$ in der zeitlichen Entwicklung nur von seinen Vorgängern abhängt, wird *markoffsch* genannt. Ist nur der jeweils letzte Vorgänger relevant, spricht man von einem *Markov-Modell erster Ordnung*. Die Zustandsübergänge eines Modells mit K Zuständen können dann durch *Übergangswahrscheinlichkeiten* $a_{ij} = P(q_t = j|q_{t-1} = i)$ mit $t = 2, .., T$ und $i, j \in \{1, .., K\}$ charakterisiert werden.

Ein HMM $\Lambda = (\boldsymbol{\pi}, \mathbf{A}, \mathbf{B})$ ist durch die $K \times K$-Matrix der Übergangswahrscheinlichkeiten

$$\mathbf{A} = [a_{ij}],$$

den Vektor der *Anfangswahrscheinlichkeiten*

$$\boldsymbol{\pi} = [\pi_i] = P(q_1 = i)$$

und (im Fall diskreter Symbole) die Matrix der *Ausgabe- bzw. Emissionswahrscheinlichkeiten*

$$\mathbf{B} = [b_{il}] = P(o_t = O_l|q_t = i)$$

spezifiziert. Letztere werden benötigt, da die Zustände nicht unmittelbar beobachtbar sind, sondern sich durch von den Zuständen abhängigen Beobachtungsfolgen äußern.

Man unterscheidet zwischen diskreten HMMs, bei denen die Beobachtungsfolgen $O = o_1..o_T$ aus Symbolen eines endlichen Alphabets $o_t \in \{O_1, .., O_L\}$ bestehen, und kontinuierlichen Modellen. Kontinuierliche HMMs sind im Wesentlichen durch die Art ihrer Emissionen in Form von Merkmalsvektoren gekennzeichnet. Abhängig vom momentan eingenommenen Zustand q_j emittiert das kontinuierliche HMM mit einer bestimmten Wahrscheinlichkeitsdichte (bzw. Emissionsdichte) b_j einen Merkmalsvektor $\mathbf{x}$. $\mathbf{B}$ ist in diesem Fall ein Vektor von Ausgabewahrscheinlichkeitsdichten.

Da die Merkmale prinzipiell jeder beliebigen Verteilung unterliegen können, werden die Emissionsdichten b_j oft mit Linearkombinationen von Normalverteilungen (in diesem Zusammenhang sogenannten Mischverteilungen) g_{jk} approximiert, so dass gilt:

$$b_j(\mathbf{x}) = \sum_{k=1}^{K_j} c_{jk} g_{jk} = \sum_{k=1}^{K_j} c_{jk} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}). \tag{2.11}$$

Die einzelnen Normalverteilungen sind charakterisiert durch ihren Mittelwert $\boldsymbol{\mu}$ und ihre Kovarianzmatrix $\boldsymbol{\Sigma}$.

2.5.1. Spezifikation und Auswertung von Modellen

Die Verwendung von HMMs beinhaltet drei Probleme:

1. **Dekodierung:** Zur Bewertung, mit welcher Wahrscheinlichkeit ein Modell eine Beobachtungsfolge generiert hat, wird die *Produktionswahrscheinlichkeit* $P(\mathbf{S}|\Lambda)$ einer Beobachtungsfolge $\mathbf{S}$ effizient mit Hilfe eines dynamischen Programms, dem *Vorwärtsalgorithmus*, bestimmt. Eine genaue Beschreibung befindet sich in Abschitt 5.2.

2. **Die wahrscheinlichste Zustandsfolge:** Gesucht ist diejenige Folge von Zuständen $\mathbf{q}^*$, die am wahrscheinlichsten eine Beobachtungsfolge $\mathbf{S}$ hervorgebracht hat. Dafür kann man wiederum ein dynamisches Programm (den *Viterbi-Algorithmus*, s. Abschitt 5.2) verwenden.

3. **Parameterschätzung:** Um die optimalen Modellparameter aus einer Menge gegebener Beobachtungsfolgen zu ermitteln, ist die *Likelihood* $L = \prod_{i=1}^{N} P(\mathbf{S}^i|\Lambda)$ (Wahrscheinlichkeit), dass die Daten durch das Modell zustande gekommen sind, zu maximieren. Dafür ist

$$\Lambda^* = \arg \max_{\Lambda=(\boldsymbol{\pi},\mathbf{A},\mathbf{B})} \prod_{i=1}^{N} P(\mathbf{S}^i|\Lambda) \tag{2.12}$$

   zu lösen. Dies geschieht unter Zuhilfenahme iterativer Verfahren durch wechselweises Berechnen der Likelihood anhand des Modells Λ und Schätzung der neuen Modellparameter $\Lambda = (\boldsymbol{\pi}, \mathbf{A}, \mathbf{B})$ durch die Auszählung der in Problem 2 ermittelten Zustandsübergänge. Dabei gibt es zwei mögliche Vorgehensweisen: Das *Baum-Welch-Training* stützt sich auf die Berechnung der vollständigen Produktionswahrscheinlichkeiten, das *Viterbi-Training* hingegen nutzt nur die optimale Zustandsfolge $\mathbf{q}^*$ als Berechnungsgrundlage und maximiert somit $\prod_{i=1}^{N} P(\mathbf{S}^i, \mathbf{q}^*|\Lambda)$. Letztere Variante ist zwar weniger rechenaufwändig, jedoch auch ungenauer.

2.5.2. Klassifikation mit HMMs

Bei der Klassifikationsaufgabe ist für jede Klasse $y_i \in Y$ ein HMM Λ_{y_i} aus einer klassifizierten Stichprobe zu schätzen. Danach können neue Beobachtungsfolgen mit Hilfe von

$$f(\mathbf{S}) = \arg\max_{y_i} P(\mathbf{S}|\Lambda_{y_i}) \tag{2.13}$$

klassifiziert werden.

2.6. Kerndichteschätzung und -klassifikation

Dichteschätzung ist ein fundamentales Problem maschinellen Lernens und aus vielen Gründen interessant. Der Bereich möglicher Anwendungen ist sehr weit. So lassen sich beispielsweise auch Klassifikationsprobleme auf Dichteschätzung zurückführen (s. Abschnitt 2.3).

Historisch betrachtet wurden Dichten zunächst durch *parametrische* Funktionen geschätzt. Verbreitet ist beispielsweise die *Maximum-Likelihood-Schätzung* von Dichten, die in vielen Situationen eine sinnvolle Approximation der Dichten darstellt. Andererseits führt ein solches Vorgehen u.U. zu einer groben Vereinfachung und ist daher nicht immer zufriedenstellend anwendbar.

Alternativen zu parametrischen Methoden sind *nicht-parametrische* Dichteschätzungen, die weniger Annahmen über die zugrundeliegenden Dichten machen. Genaugenommen wird nur die Annahme gemacht, dass den Beispieldaten eine Dichte zugrundeliegt: „..the data will be allowed to speak for themselves“ (s. [Sil86]). Es existiert eine Vielzahl von nicht-parametrischen Dichteschätzern, unter anderem sind Histogramme, kernbasierte Verfahren und Nächster-Nachbar-Schätzer zu nennen. Gute Übersichten zu diesem Thema sind in [TT78], [Fre77] und [Sil86] zu finden. Nicht-parametrische Schätzverfahren benötigen eine höhere Rechenkapazität und haben daher erst in den letzten Jahrzehnten eine weite Verbreitung gefunden. Mittlerweile sind sie aber recht gut erforscht und bilden eine wichtige Grundlage der modernen statistischen Datenanalyse.

In dieser Arbeit konzentrieren wir uns auf Kerndichteschätzer (*Kernel Density Estimator*, KDE), die ein einfaches und gut interpretierbares Mittel zur nicht-parametrischen Dichteschätzung darstellen.

2.6.1. Dichtekerne

Dichtekerne sind einfache Glättungsmethoden und finden Anwendungen in diversen Applikationen des statistischen und des maschinellen Lernens, wie Datenvisualisierung, Regression, Klassifikation und Clusteranalyse.

Dichtekerne sind Abbildungen $K : \mathbb{R}^d \mapsto \mathbb{R}$, für die üblicherweise gilt:

1. **Normeigenschaft:** $\int_{-\infty}^{\infty} K(\mathbf{x}) d\mathbf{x} = 1$,
2. **Positivität:** $K(\mathbf{x}) \geq 0$.

Diese Eigenschaften stellen sicher, dass die Gesetze der Wahrscheinlichkeitstheorie nicht verletzt werden.

Etablierte Kerne sind beispielsweise der

Epanechnikovkern: $K(x) = \frac{3}{4}(1 - \frac{1}{5}x^2)/\sqrt{5}$ für $x < 5$,

Dreieckskern: $K(x) = \begin{cases} (1 - |x|) & \text{für } |x| < 1, \\ 0 & \text{sonst,} \end{cases}$

Gausskern: $K(x) = \frac{1}{\sqrt{2\pi}} e^{-(1/2)x^2}$.

2.6.2. Kerndichteschätzer

Kerndichteschätzer funktionieren nach dem Prinzip der Mittelwertbildung:

$$\hat{p}(\mathbf{x}) = \sum_{i=1}^{N} \omega_i K\left(\frac{\mathbf{x}-\mathbf{x}_i}{l}\right) \tag{2.14}$$

$$\text{mit } \omega_i = \frac{1}{N}. \tag{2.15}$$

Der Abstand jedes Beispieldatums zum gesuchten $\mathbf{x}$ wird über die Kernfunktion $K(.)$ bewertet und anteilig aufsummiert.

Der Parameter l bezeichnet dabei die *Fensterbreite* (auch *Glättungsparameter* oder *Bandbreite* genannt). Dieser hat einen großen Einfluss auf die Güte der Dichteschätzung. Je größer l wird, desto größer wird der Einfluss von entfernten $\mathbf{x}_i$.

Es existiert eine Vielzahl von Veröffentlichungen, die sich mit der Frage der geeigneten Kern- und Fensterbreitenwahl beschäftigen (beispielsweise [Sil86], [Tur] oder [Duo04]).

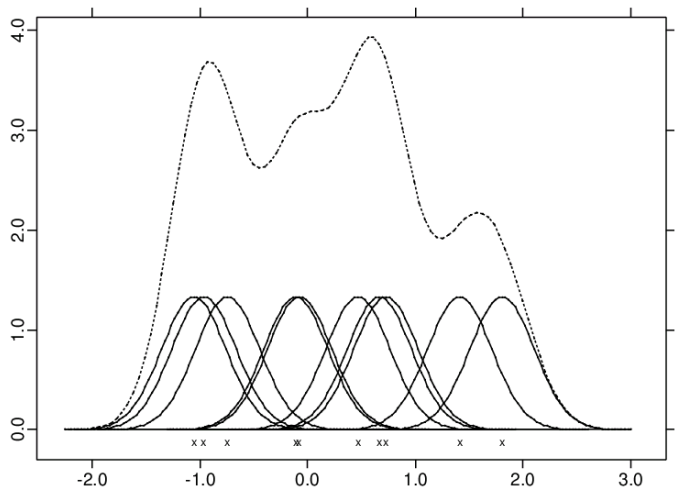

Abbildung 2.3.: Univariater Kerndichteschätzer $\hat{p}(x)$ (gestrichelte Linie) als Summe der $\frac{1}{N}K((x-x_i)/l)$ (durchgehende Linie) (aus [Tur]).

2.6.3. Kerndichteklassifikatoren

Mit den geschätzten Dichten eines KDEs lassen sich nun einfache Klassifikatoren, die *Kerndichteklassifikatoren* (*Kernel Density Classifier*, KDC), realisieren.

Der in Abschnitt 2.3 beschriebene Bayesklassifikator klassifiziert nach

$$f(\mathbf{x}) = \arg\max_{y_i} P(y_i|\mathbf{x}) = \arg\max_{y_i} p(\mathbf{x}|y_i)P(y_i). \tag{2.16}$$

Ein KDE kann herangezogen werden, um die klassenspezifischen Dichten $p(\mathbf{x}|y_i)$ zu

schätzen. Die a-priori Wahrscheinlichkeiten der Klassen werden häufig durch

$$P(y_i) = \frac{N_{y_i}}{N} \tag{2.17}$$

approximiert. Allerdings ist keineswegs sicher, dass die Häufigkeit der einzelnen Klassen durch die Häufigkeit der Trainingsbeispiele wiedergegeben wird (die Art der Datenerhebung kann beispielsweise zugunsten einer Klasse verschoben sein), sodass die Annahme a-priori-gleichwahrscheinlicher Klassen, unabhängig von der Anzahl der bekannten Beispiele, in der Praxis ebenso sinnvoll sein kann:

$$P(y_i) = \frac{1}{M}. \tag{2.18}$$

2.7. Support-Vektor-Maschinen

Support-Vektor-Maschinen (*Support Vector Machines*, SVMs) sind leistungsfähige binäre Klassifikatoren, welche Mitte der Neunziger Jahre auf der Basis neuerer Erkenntnisse der statistischen Lerntheorie (vgl. [Vap95], [Vap98]) entwickelt wurden. Die Idee dahinter ist, die *Kapazität* einer zu lernenden Trennfunktion so zu beschränken, dass trotz guter Klassifikationsleistung auf den Trainingsdaten die Generalisierungsfähigkeit erhalten bleibt. Das durch eine SVM konstruierte Modell bleibt dennoch mathematisch gut analysierbar, da die Trennfunktion auch im nichtlinearen Fall stets eine Hyperebene ist, welche in einem (u. U. unendlichdimensionalen) erweiterten Merkmalsraum liegt. Dabei wird diese meist rechenaufwändige Transformation in den *Feature Space* nicht explizit berechnet, sondern implizit mittels des sogenannten *Kernel-Tricks* durchgeführt. Eine gute und ausführliche Einführung zu SVMs findet sich in [Bur98] sowie [CS00].

Der Einsatz von SVMs ist weit verbreitet und hat sich in einigen Domänen des maschinellen Lernens bereits zu einem Standardwerkzeug entwickelt. So erfreut sich die SVM im Data Mining großer Beliebtheit. Auch in der Bild- und Objekterkennung (z.B. Handschrifterkennung) werden SVMs erfolgreich eingesetzt. Weitere Anwendungen im Feld der Mustererkennung betreffen die Gebiete Sprach- und Gesichtserkennung. In der Signalverarbeitung haben sich SVMs bei der seismischen Signalklassifikation und der Dichteschätzung (z.B. in der Geologie) bewährt. Schließlich werden in der Bioinformatik SVMs dazu benutzt, um DNA- bzw. Proteinsequenzen zu klassifizieren (bzw. auf Homologien zu untersuchen) und Proteinstrukturen vorherzusagen. Eine ständig aktualisierte Liste der Anwendungsmöglichkeiten hält [Guy05] bereit.

2.7.1. Strukturelle Risikominimierung

Die *statistische Lerntheorie* untersucht, welche Aussagen über die generelle Lernbarkeit einer Datenmenge gemacht werden können. Ein wichtiges Resultat in diesem Zusammenhang ist die Obergrenze der Generalisierungsfähigkeit, welche unter einem gegebenen Konfidenzintervall (z.B. 95%-ige Sicherheit $\rightarrow \eta = 0,05$) mit der Wahrscheinlichkeit

$(1-\eta)$ den tatsächlichen Fehler nach oben abschätzt durch:

$$R(\mathbf{w}) \leq R_{emp}(\mathbf{w}) + \Phi(h, N, \eta) \tag{2.19}$$

mit der sogenannten Vapnik-Chervonenkis-(VC-)Konfidenz

$$\Phi(h, N, \eta) = \sqrt{\frac{h(\log \frac{2N}{h} + 1) - \log(\frac{\eta}{4})}{N}}. \tag{2.20}$$

N bezeichnet die Anzahl der Trainingsbeispiele und h die VC-Dimension des verwendeten Hypothesenraums. R_{emp} ist das *empirische Risiko*, das durchschnittliche Risiko über die Trainingsbeispiele

$$R_{emp}(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} L(f(\mathbf{x}_i, \mathbf{w})) \tag{2.21}$$

bzgl. einer Verlustfunktion L (vgl. [CM98]). Die VC-Dimension einer Menge von Funktionen $\{f(\mathbf{w})\}$ ist definiert als die maximale Anzahl von Trainingsbeispielen, die durch diese Funktionenklasse in allen Konstellationen separiert werden können und liefert damit ein Maß für deren *Kapazität*. Um eine gute Generalisierungsfähigkeit zu erhalten, ist diese möglichst klein zu halten. Für die Funktionenklasse der linearen Trennfunktionen im $\mathbb{R}^d$ z.B. ist $h_{Lin} = d + 1$.

Bei der *empirischen Risikominimierung* wählt man ein Modell aus der Modellmenge aus, welches den rechten Term der Gleichung 2.19 minimiert und somit einen Kompromiss zwischen empirischem Fehler und (der mit 2.20 wachsenden) VC-Dimension eingeht. Die *strukturelle Risikominimierung* hingegen nutzt die Zerlegung der Schranke des Generalisierungsfehlers und erlaubt es, auf strukturierten Modellmengen

$$\mathcal{M} = \bigcup_i \mathcal{M}_i \tag{2.22}$$

mit $\mathcal{M}_1 \subseteq \mathcal{M}_2 \subseteq \cdots$ und $h_{\mathcal{M}_1} \leq h_{\mathcal{M}_2} \leq \cdots$ Modelle mit minimalem Risiko für eine bestimmte Funktionenklasse zu trainieren.

Für *linear separierbare Probleme* – d.h. für Daten, die durch eine lineare Trennfunktion ohne Fehler klassifizierbar sind – kann man eine Funktionenklasse angeben, deren Kapazität berechnet werden kann. Die Klasse der *Hyperebenen*

$$\mathcal{H} = \{\mathbf{x} | \mathbf{w} \cdot \mathbf{x} + b = 0\} \tag{2.23}$$

mit $\mathbf{w}, \mathbf{x} \in \mathbb{R}^d, b \in \mathbb{R}$ und der Entscheidungsfunktion

$$f(\mathbf{x}) = sgn(\mathbf{w} \cdot \mathbf{x} + b) \tag{2.24}$$

für ein 2-Klassen-Problem, hat im Allgemeinen mehrere Lösungen. Die Kapazität einer Hyperebene ist umso kleiner, je größer ihr Abstand zu den Datenpunkten beider Klassen ist.

2.7.2. Die optimal separierende Hyperebene

Ausgehend von den oben aufgeführten Überlegungen erhält man die *optimal separierende Hyperebene*, wenn der Korridor bzw. Rand (*margin*, s. Abb. 2.4) dieser möglichst breit wird. Die Maximierung des Abstands aller Trainingsbeispiele zur Hyperebene führt auf die Minimierung des Normalenvektors $\mathbf{w}$ der *kanonischen Hyperebene*:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 \tag{2.25}$$

mit der Bedingung

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \quad \forall i = 1, .., N. \tag{2.26}$$

Die Bedingung in Gleichung 2.26 stellt dabei sicher, dass alle Datenpunkte mindestens den Abstand 1 von der Hyperebene haben. Der Rand der Hyperebene beträgt

$$\mathbf{w} \cdot (\mathbf{x}_+ - \mathbf{x}_-) = \frac{2}{\|\mathbf{w}\|}.$$

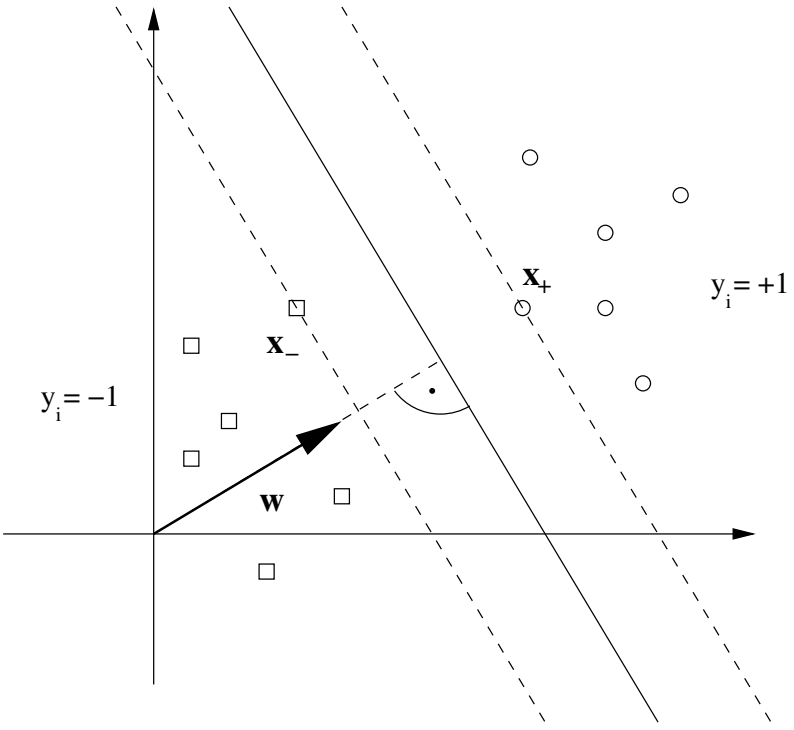

Abbildung 2.4.: Veranschaulichung der Funktionsweise einer Support-Vektor-Maschine anhand eines 2-Klassen-Beispielproblems. Die durchgezogene Linie verkörpert die optimal separierende Hyperebene, die gestrichelten Linien stellen die Ränder $\{\mathbf{x}|\mathbf{w} \cdot \mathbf{x} + b = \pm 1\}$ dar.

Optimierung

Die Minimierung der *Zielfunktion* 2.25 stellt ein quadratisches Optimierungsproblem mit linearen Nebenbedingungen dar. Dazu ist die *Lagrange-Funktion* (vgl. [SS02])

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i (y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1) \tag{2.27}$$

mit den *Lagrange-Multplikatoren* $\alpha_i \geq 0$ nach $\mathbf{w}$ und b zu minimieren und nach α_i zu maximieren. Dies entspricht der Bestimmung ihres Sattelpunktes. Mit dem Einsetzen der Bedingungen aus der 1. Ableitung $\sum_{i=1}^{N} \alpha_i y_i = 0$ und $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$ in 2.27 erhält man das *duale Problem*

$$\max_{\boldsymbol{\alpha}} \quad W(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{2.28}$$

$$\text{u.d.B.d.} \quad \alpha_i \geq 0 \text{ und } \sum_{i=1}^{N} \alpha_i y_i = 0. \tag{2.29}$$

Die Entscheidungsfunktion kann jetzt mittels

$$f(\mathbf{x}) = sgn\left(\sum_{i=1}^{N} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}) + b\right) \tag{2.30}$$

dargestellt werden.

Diejenigen $\mathbf{x}_i$ mit $\alpha_i > 0$ werden *Supportvektoren* (SV) genannt und liegen auf dem Rand der Hyperebene. Die Lösung hängt allein von ihnen ab, d.h. man erhält bei einigen $\alpha_i = 0$ eine *spärliche* Repräsentation der erforderlichen Trainingsdatenmenge.

Für die Optimierung existieren effiziente Lösungsmöglichkeiten und Techniken, z.B. *SVMlight* ([Joa99]) oder *Sequential Minimal Optimization* (SMO, [Pla98]).

2.7.3. Lineare Support-Vektor-Klassifikation

Für zwei linear separable Klassen $y_i \in \{1, -1\}$ kann nun nach dem oben geschilderten Prinzip eine optimal separierende Hyperebene bestimmt werden. Für die Klassifikation eines neuen Beispiels $\mathbf{x}_i$ wird Gleichung 2.30 benutzt, wobei bei positivem Ergebnis Klasse 1 ($y_i = 1$) und bei negativem Ergebnis Klasse 2 ($y_i = -1$) gewählt wird.

Lineare Trennung nicht linear separierbarer Daten

Sind die Trainingsbeispiele nicht linear separabel, existiert keine separierende Hyperebene nach Gleichungen 2.25 und 2.26. Dieses Problem kann gelöst werden, indem eine Verletzung der Bedingung 2.26 durch die Einführung von *Hilfsvariablen* (*slack variables*,

s. [SS02]) zugelassen wird:

$$\xi_i \geq 0 \quad \forall i = 1, .., N. \tag{2.31}$$

In den Nebenbedingungen muss dann

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i \quad \forall i = 1, .., N \tag{2.32}$$

erfüllt sein. Um die Lernfähigkeit der SVM aufrecht zu erhalten, müssen die Hilfsvariablen ebenfalls in der Zielfunktion berücksichtigt werden:

$$\min_{\mathbf{w},\boldsymbol{\xi}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i. \tag{2.33}$$

Für das duale Problem ändern sich lediglich die Nebenbedingungen zu

$$0 \leq \alpha_i \leq C. \tag{2.34}$$

Der Faktor C bestimmt den Kompromiss zwischen der Randmaximierung und der Trainingsfehlerminimierung und ist durch eine Evaluation (z.B. Kreuzvalidierung, s. Abschnitt 2.9) zu optimieren.

2.7.4. Nichtlineare Klassifikation

Eine weitere Möglichkeit, linear nicht separable Daten zu trennen, besteht darin, eine nichtlineare Transformation dieser vorzunehmen. Dies erhöht zugleich die Kapazität der SVM und kann eine adäquatere Repräsentation der Daten bedeuten.

Erweiterung und Transformation des Merkmalsraumes

Ähnlich wie beim Polynomklassifikator (s. [SK91]) wird der Merkmalsraum durch eine Abbildung

$$\Phi : \mathbb{R}^d \rightarrow \mathbb{R}^D \tag{2.35}$$

erweitert, wobei üblicherweise $D > d$ gilt. Dazu sind in den Gleichungen 2.28 und 2.30 $\mathbf{x}$ bzw. $\mathbf{x}_i$ durch $\Phi(\mathbf{x})$ bzw. $\Phi(\mathbf{x}_i)$ zu ersetzen. Die Transformation $\Phi(\mathbf{x})$ kann allerdings im Gegensatz zum Polynomklassifikator auch andere Funktionenklassen als Polynome umfassen. Ziel ist die lineare Separierbarkeit der Daten im erweiterten Merkmalsraum.

Der Kernel-Trick

Die explizite Transformation der Merkmalsvektoren $\mathbf{x}$ in einen höherdimensionalen Merkmalsraum ist mit einem Mehraufwand für die Berechnung von $\Phi(\mathbf{x})$ sowie für die Optimierung verbunden. Die genaue Betrachtung der Gleichungen 2.28 und 2.30 offenbart, dass die Datenbeispiele jeweils nur in Skalarprodukten auftreten. Durch Anwendung des *Kernel-Tricks* (vgl. [SS02]), einer impliziten Transformation der Merkmalsvektoren in einen erweiterten Merkmalsraum, kann der Mehraufwand u.U. reduziert werden. Hierfür

werden die Skalarprodukte in den beiden Gleichungen durch eine *Kernfunktion* (*kernel function*) $k(\cdot, \cdot)$ ersetzt, sodass

$$\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}_i) = k(\mathbf{x}, \mathbf{x}_i). \tag{2.36}$$

Dabei repräsentiert die Kernfunktion im Unterschied zu den Dichtekernen in Abschnitt 2.6 ein inneres Produkt zweier Vektoren. Daher unterliegen diese Kernfunktionen anderen, namentlich den Mercer-Bedingungen, welche verlangen, dass die $N \times N$-Kernmatrix (Gram-Matrix) $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ positiv semidefinit ist. Valide Kernfunktionen sind z.B. der

- **Lineare Kern:** $k(\mathbf{x}, \mathbf{x}_i) = \mathbf{x} \cdot \mathbf{x}_i$,
- **Polynomkern vom Grad** g**:** $k(\mathbf{x}, \mathbf{x}_i) = (\mathbf{x} \cdot \mathbf{x}_i)^g$,
- **RBF- bzw. Gausskern:** $k(\mathbf{x}, \mathbf{x}_i) = e^{(-\frac{\|\mathbf{x} \cdot \mathbf{x}_i\|}{\gamma})}$ mit der Kernbreite γ.

Die optimalen Parameter g bzw. γ sind wiederum durch eine Evaluation zu bestimmen.

Schließlich kann man die beiden Techniken (Hilfsvariablen und Kernel-Trick) kombinieren, um selbst in diesem hochdimensionalen Raum die Klassifikationsleistung zu verbessern. Dies erfordert jedoch die gleichzeitige Evaluation bzgl. zweier Hyperparameter.

2.7.5. Multiklassenklassifikation

Bisher haben wir nur den 2-Klassen-Fall besprochen, in der Realität treten aber viele Probleme mit mehr als zwei Klassen auf. Aus jedem Multiklassenproblem ($M > 2$) lassen sich jedoch mehrere 2-Klassen-Probleme machen.

Eine Möglichkeit besteht darin, für jede Klasse einen Klassifikator gegen die zusammengefassten restlichen Trainingsbeispiele zu trainieren. Diese Strategie wird 1-gegen-Alle (*one-against-all*) genannt. Insgesamt gibt es M solcher binärer Klassifikatoren.

Eine andere Variante ist die paarweise Klassifikation (1-gegen-1, *one-against-one*), bei der $\frac{M(M-1)}{2}$ Klassifikatoren aller möglichen Klassenpaarkonstellationen trainiert werden. Für große M ist diese Strategie aufgrund des Rechenaufwandes jedoch nicht verwendbar.

2.7.6. Ein-Klassen-SVM

Im unüberwachten Lernfall – d.h. wenn keine Kategoriebezeichnungen oder negativen Trainingsbeispiele vorliegen bzw. wenn klassenweise separat trainiert wird – kann man die SVM dahingehend modifizieren, dass lediglich das „Volumen “ der vorliegenden Daten geschätzt wird (vgl. [SS02]). Einsatzgebiet dieses Verfahrens ist z.B. die Ausreißerdetektion.

Die Strategie besteht darin, die Daten (ggf. nach der Transformation in den erweiterten Merkmalsraum) mit maximalem Rand vom Ursprung zu separieren. Die Zielfunktion für

N Datenbeispiele lautet

$$\min_{\mathbf{w},\xi,\rho} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{\nu N}\sum_{i=1}^{N}\xi_i - \rho \tag{2.37}$$

$$\text{u.d.B.d} \quad (\mathbf{w}\cdot\Phi(\mathbf{x}_i)) \geq \rho - \xi_i, \quad \xi_i \geq 0, \tag{2.38}$$

wobei der Parameter $\nu \in]0,1]$ die maximale relative Anzahl von Supportvektoren bzw. die minimale relative Anzahl zu berücksichtigender Ausreißer steuert. Je näher ν dabei an 0 liegt, desto kleiner muss der Abstand zum Ursprung ρ werden. Dies wird *hard margin*-Fall genannt, da die Bestrafung der Fehler gegen ∞ geht. Aus den Gleichungen 2.37 und 2.38 ergibt sich folgendes duales Problem:

$$\max_{\boldsymbol{\alpha}} \quad \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j k(\mathbf{x}_i,\mathbf{x}_j) \tag{2.39}$$

$$\text{u.d.B.d.} \quad 0 \leq \alpha_i \leq \frac{1}{\nu N} \text{ und } \sum_{i=1}^{N}\alpha_i = 1. \tag{2.40}$$

2.8. Hauptkomponentenanalyse

Die Hauptkomponentenanalyse (*principal component analysis*, PCA, s. [Mit97], [CM98]) wurde in den 30er Jahren von Harold Hotelling eingeführt. Das Ziel der PCA ist die Projektion eines hochdimensionalen Raumes in einen latenten (Unter-)Raum, der durch Charakteristika der gegebenen Daten bestimmt wird. Gesucht wird ein orthogonaler Raum, in dem die Daten in Richtung der maximalen Varianzen (den sogenannten *Hauptrichtungen*) repräsentiert werden. Die PCA geht also davon aus, dass Richtungen mit hoher Varianz Richtungen mit hohem Informationsgehalt entsprechen, was allerdings nicht immer zutreffen muss.

Anwendungen findet die PCA in der Datenvisualisierung und -kompression, in der Merkmalsextraktion und -reduktion für Algorithmen der Mustererkennung.

2.8.1. Algorithmisches Vorgehen

Sei $\mathbf{x} = (x_1,..,x_d)^T$ eine n-dimensionale Zufallsvariable und $\mathbf{x}' = (x'_1,..,x'_m)^T$ die gesuchte Transformierte. Dann berechnet die PCA eine $d \times m$-Transformationsmatrix $\mathbf{W}$, für die $\mathbf{x}' = \mathbf{W}\mathbf{x}$ unter den gegebenen Bedingungen gilt.

Um die Hauptrichtungen einer Spaltenmatrix von Beispieldaten $\mathbf{X} = [\mathbf{x}_1,..,\mathbf{x}_N]$ zu schätzen, muss zunächst die Kovarianzmatrix $\mathbf{C}$ der mittelwertzentrierten Daten berechnet werden. O.b.d.A. soll gelten:

$$\mathbf{X} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i = 0. \tag{2.41}$$

Die geschätzte Kovarianzmatrix ergibt sich dann zu

$$\hat{\mathbf{C}} = \frac{1}{N}\mathbf{X}^T\mathbf{X}. \tag{2.42}$$

Die Kovarianzmatrix ist symmetrisch und positiv definit.
Eine Eigenvektorzerlegung der Kovarianzmatrix ergibt

$$\hat{\mathbf{C}} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T, \tag{2.43}$$

wobei $\mathbf{V}$ die Eigenvektoren $(\mathbf{v}_1, .., \mathbf{v}_d)$ als Spalten beinhaltet und $\mathbf{\Lambda} = diag(\lambda_1, .., \lambda_d)$ eine aus den Eigenwerten der Kovarianzmatrix gebildete Diagonalmatrix ist. Der Eigenvektor $\mathbf{v}_i$ entspricht der i-ten gesuchten Hauptkomponente und der zugehörige Eigenwert λ_i der Varianz dieser Hauptkomponente. Im Folgenden gilt o.B.d.A. $\lambda_1 > \lambda_2 > ... > \lambda_d$.
Die Projektion der Ausgangsdaten in den gesuchten Raum ist durch

$$\mathbf{X}' = \mathbf{V}\mathbf{X} \tag{2.44}$$

möglich.

Zur Dimensionsreduktion werden nur die Eigenwerte bis zu einem $\lambda_j,\ j < d$ zur Projektion herangezogen. Es bietet sich zum Beispiel an, zu Visualisierungszwecken die ersten beiden Hauptrichtungen zu betrachten. Zur Datenkompression ist es sinvoll, all die Richtungen zu betrachten, die einen Großteil der Gesamtvarianz (beispielsweise $> 90\%$) auf sich vereinen. Unter der Annahme, dass die Varianz dem Informationsgehalt entspricht, können so die wenig informativen Richtungen erkannt und ohne großen Verlust eliminiert werden.

In der Praxis findet der oben beschriebene Algorithmus allerdings selten Verwendung. Die Berechnung der Kovarianzmatrix ist aufwändig und speicherintensiv und daher für hochdimensionale Räume nicht mehr durchführbar. Hier bietet die Singulärwertzerlegung eine Alternative.

Eine Datenmatrix $\mathbf{X}$ der Dimensionalität $N \times d$ mit $d < N$ lässt sich in

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T \tag{2.45}$$

zerlegen. Die Spaltenvektoren von $\mathbf{U} \in \mathbb{R}^{d\times N}$ und $\mathbf{V} \in \mathbb{R}^{d\times d}$ sind paarweise orthogonal, so dass $\mathbf{U}^T\mathbf{U} = \mathbf{I}_d$ eine Identitätsmatrix der Dimensionalität d ist. $\mathbf{S} = diag(s_1, .., s_d)$ ist eine Diagonalmatrix von Singulärwerten. Es gilt

$$N\mathbf{C} = \mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{S}\mathbf{U}^T\mathbf{U}\mathbf{S}\mathbf{V}^T \tag{2.46}$$

$$= \mathbf{V}\mathbf{S}^2\mathbf{V}^T \tag{2.47}$$

$$\Leftrightarrow \mathbf{C} = \frac{1}{N}\mathbf{V}\mathbf{S}^2\mathbf{V}^T. \tag{2.48}$$

Weiterhin gilt

$$\lambda_i = \frac{1}{N}s_i^2, \tag{2.49}$$

womit der Zusammenhang zwischen Singulärwert- und Eigenwertzerlegung der Kovarianzmatrix unmittelbar deutlich wird. Insbesondere ist hervorzuheben, dass die i-te Spalte von $\mathbf{US}$ die Werte der i-ten Hauptkomponente enthält.

Für den interessanten Fall $N < d$ lassen sich durch Transponieren die Plätze von $\mathbf{V}$ und $\mathbf{U}$ vertauschen.

$$\mathbf{X}^T = \mathbf{USV}^T = \tilde{\mathbf{V}}\mathbf{S}\tilde{\mathbf{U}}^T \tag{2.50}$$

Die Rollen von $\mathbf{V}$ und $\mathbf{U}$ bleiben erhalten, $\mathbf{V}$ enthält lediglich die ersten N Eigenvektoren von $\mathbf{X}^T\mathbf{X}$ als Spaltenvektoren.

2.9. Kreuzvalidierung

Generalisierungsperformanz

Die Generalisierungsperformanz (bzw. -fähigkeit) eines lernenden Systems ist von größter Wichtigkeit, um die Voraussagequalität unbekannter Daten einschätzen zu können. In der Praxis benötigen wir diese Performanz einerseits, um verschiedene lernende Systeme miteinander zu vergleichen, also zur Methodenwahl, andererseits um das geeignete Modell bzw. die geeigneten Hyperparameter einer Methode zu wählen (Modellselektion). Es muss sichergestellt werden, dass das Modell die Daten nicht auswendig lernt, ein Phänomen, dass unter dem Namen *overfitting* bekannt ist. Typischerweise führt dies zu einer guten Trainingsperformanz, aber einer ausserordentlich schlechten Performanz auf unbekannten Daten (vgl. [HTF01]).

Der *Testfehler* ist der Erwartungswert über unbekannte Datenbeispiele

$$E = E\{L(f(\mathbf{x}), y_i)\}, \tag{2.51}$$

der sich leicht durch

$$\hat{E} = \frac{1}{N}\sum_{i=1}^{N} L(f(\mathbf{x}_i), y_i) \tag{2.52}$$

schätzen lässt. Das Ziel ist, diesen erwarteten Fehler zu minimieren.

Kreuzvalidierung

Eine einfache, verbreitete Methode dieses Ziel zu erreichen, ist die *Kreuzvalidierung* (*cross validation*, CV). Die Trainingsdaten werden zufällig in F ungefähr gleich große Partitionen (*Folds*) aufgeteilt. Ein Fold g fungiert als Testmenge und mit den übrigen $F-1$ Folds wird das Modell f trainiert und anschließend der Testfehler der zurückgehaltenen Testmenge berechnet. Dieses Vorgehen wird mit allen F Folds wiederholt. Danach wird der Gesamttestfehler über die Testfehler der Folds gemittelt. Für $F = 3$ stellt sich das Szenario wie in Abbildung 2.5 dar.

Formal kann die Kreuzvalidierung wie folgt beschrieben werden: Sei $\kappa : \{1, .., N\} \mapsto \{1, .., F\}$ eine Indexabbildung, die zufällig eine Beobachtung $i \in \{1, .., N\}$ einem Fold $g \in \{1, ..F\}$ zuordnet und sei $\hat{f}^{-\kappa}(\mathbf{x})$ ein Modell, das ohne den g-ten Fold der Daten

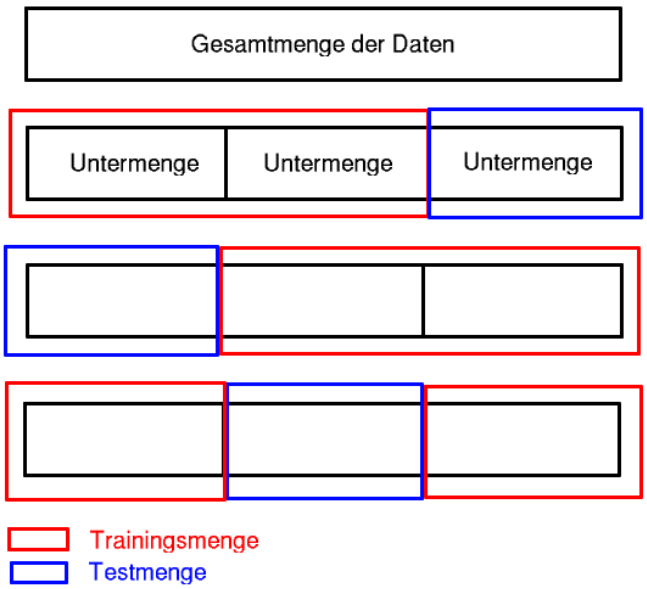

Abbildung 2.5.: 3-Fold Kreuzvalidierung: Schematische Darstellung der Datenpartitionierung.

trainiert wurde, so ist der von der Kreuzvalidierung geschätzte Fehler

$$\hat{E}_{CV} = \frac{1}{N} \sum_{i=1}^{N} L(\hat{f}^{-\kappa(i)}(\mathbf{x}_i), y_i). \tag{2.53}$$

Die Wahl von F

Die Wahl der Anzahl der Partitionen F hat großen Einfuss auf die Güte der Schätzung von $\hat{E}_{CV}$. Große F, etwa $F = N$ (auch als *leave-one-out cross validation* bekannt), erreichen eine hohe Verlässlichkeit in der Qualität der Schätzung von $\hat{E}_{CV}$, sind allerdings, je nach verwendeter Methode, unter Umständen sehr rechenintensiv. Das andere Extremum $F = 2$ ist verhältnismäßig schnell berechenbar, dafür aber sehr ungenau. In der Praxis haben sich $F = 5$ und $F = 10$ als gute Kompromisse etabliert (siehe [HTF01]).

3. Alignment - Methoden

3.1. Dynamic Time Warping

Die Technik der dynamischen Zeitverzerrung (*Dynamic Time Warping*, DTW) wurde erstmals mit Erfolg in der Spracherkennung eingesetzt (s. [SC78]), um das Problem unterschiedlich langer Folgen von Merkmalsvektoren und zeitlichen Variationen in diesen Sequenzen (in diesem Fall ganze Wörter) zu lösen. Die mittels DTW erzielten Alignments von Referenz- und Testmustersequenzen werden als Distanzmaß im Rahmen eines Prototypenklassifikators (s. Abschnitt 2.4) benutzt, um vorher gespeicherte sprachliche Äußerungen zu erkennen.

Die Summe der euklidischen Abstände der jeweiligen Merkmalsvektoren gleichen Zeitindizes eignet sich nicht als Distanzmaß zweier solcher Sequenzen, da – über das notwendige Abschneiden der längeren Sequenz hinaus – die euklidische Distanz sehr anfällig gegenüber kleinen Abweichungen auf der Zeitachse ist. Das Prinzip der dynamischen Zeitverzerrung besteht darin, die beiden unterschiedlichen Zeitachsen zweier Sequenzen mittels Minimierung eines geeigneten globalen Abstandsmaßes in einen Zuordnungs- bzw. Verzerrungspfad (*warping path*) zu transformieren (s. Abb. 3.1).

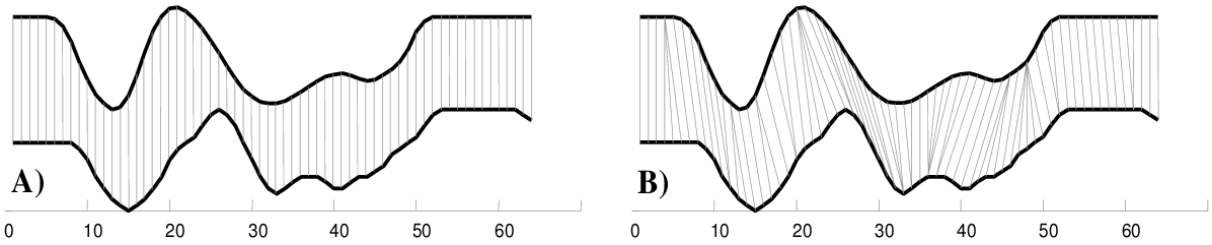

Abbildung 3.1.: Vergleich einer linearen (A) und einer durch DTW berechneten Zuordnung (B). Die Linien zwischen den Sequenzen stellen die Indizes des Verzerrungspfades dar (aus [KP99]).

3.1.1. Der DTW-Algorithmus

Mit den aus Merkmalsvektoren bestehenden Zeitserien (Sequenzen) $\mathbf{R} = \mathbf{r}_1..\mathbf{r}_m$ und $\mathbf{S} = \mathbf{s}_1..\mathbf{s}_n$ ergibt sich als Zuordnungsfunktion eine $m \times n$-Matrix

$$\mathbf{\Phi} : \{1, .., T\} \rightarrow \{1, .., m\} \times \{1, .., n\}, \tag{3.1}$$

wobei T die Länge des Verzerrungspfades $\mathbf{w} = w_1..w_T,\ \max(m, n) \leq T \leq m + n - 1$ ist. Dabei ist $w_t = (i, j)_t$ ein Indexpaar der als $m \times n$-Matrix notierbaren paarweisen Distan-

zen $d(\mathbf{r}_i, \mathbf{s}_j)$ zweier Merkmalsvektoren. Als Distanzmaß eignet sich hier die euklidische Distanz $d(\mathbf{r}_i, \mathbf{s}_j) = \|\mathbf{r}_i - \mathbf{s}_j\|$.

Der Verzerrungspfad muss dabei üblicherweise folgende Bedingungen erfüllen:

- **Anfangs- und Endbedingung:** $w_1 = (1, 1)$ und $w_T = (m, n)$.
- **Kontinuitätsbedingung:** Gegeben $w_{T-1} = (a, b)$, $w_T = (a', b')$; dann ist $a' - a \leq 1$ und $b' - b \leq 1$.
- **Monotoniebedingung:** Gegeben $w_{T-1} = (a, b)$, $w_T = (a', b')$; dann ist $a' - a \geq 0$ und $b' - b \geq 0$. Dies sichert eine ordnungserhaltende Abbildung.

Trotz dieser Einschränkungen gibt es immer noch eine exponentiell mit m und n wachsende Anzahl von möglichen Pfaden. Wir sind jedoch nur an dem Pfad interessiert, welcher die Verzerrungskosten (*warping costs*)

$$D(\mathbf{R}, \mathbf{S}) = \frac{1}{T} \sum_{t=1}^{T} w_t \tag{3.2}$$

minimiert.

Dies lässt sich unter den gegebenen Bedingungen sehr effizient mit Hilfe der dynamischen Programmierung lösen. Dazu stellen wir eine vom letzten Element ausgehende rekursive Gleichung auf:

$$\gamma(i,j) = \begin{cases} d(\mathbf{r}_1, \mathbf{s}_1), & i = j = 1 \\ d(\mathbf{r}_i, \mathbf{s}_j) + min\{\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\}, & i > 1, j > 1 \\ \infty, & \text{sonst.} \end{cases} \tag{3.3}$$

Hierbei ist $\gamma(i, j)$ die kumulative Distanz des bisher verfolgten Pfades. Um den optimalen Pfad zu erhalten, müssen die jeweiligen Zuordnungen (i, j) in der Zuordnungsfunktion $\mathbf{\Phi}$ gespeichert werden.

Der Rechenaufwand, um zwei solche Sequenzen zu alignieren beträgt $O(mn)$. Es existieren jedoch Verfahren zur Beschleunigung (s. z.B. [KP99], [CKHP02]), außerdem ist es möglich, nur wenige aussichtsreiche Pfade nahe des optimalen Pfades weiter zu verfolgen und den Suchweg damit zu beschneiden (*pruning*).

3.2. Paarweises Alignment

Paarweise Alignments bewerten den Zusammenhang zwischen *zwei* Sequenzen. Die Sequenzen werden mit dem Ziel gegeneinander ausgerichtet, möglichst viele identische oder ähnliche Sequenzpositionen in den Sequenzen oder in Teilsequenzen zu finden (vgl. Abb. 3.2). Identität (*Identity*) bezeichnet die Anzahl der Positionen im Alignment, die gleich sind, Ähnlichkeit (*similarity*) wird anhand von Ähnlichkeitsmatrizen oder auch Substitutionsmatrizen berechnet. Beide Werte sind quantitativ und werden in der bioinformatischen Sequenzanalyse als Kriterium für Homologie herangezogen.

```
Sequenz 1: THEFA_TCAT
           ||||| ||||
Sequenz 2: THEFASTCAT
```

Abbildung 3.2.: Mögliches paarweises Alignment zweier Sequenzen.

3.2.1. Punktdiagramme

Das einfachste paarweise Alignment ist ein Punktdiagramm (*dot plot*). Zwei Sequenzen werden sich an den Seiten einer Matrix gegenübergestellt und alle identischen Sequenzpositionen werden in der Matrix duch einen Punkt bzw. eine Line markiert (s. Abb. 3.3). Zwei identische Sequenzen haben eine durchgehend markierte Linie auf der Diagonalen der Matrix.

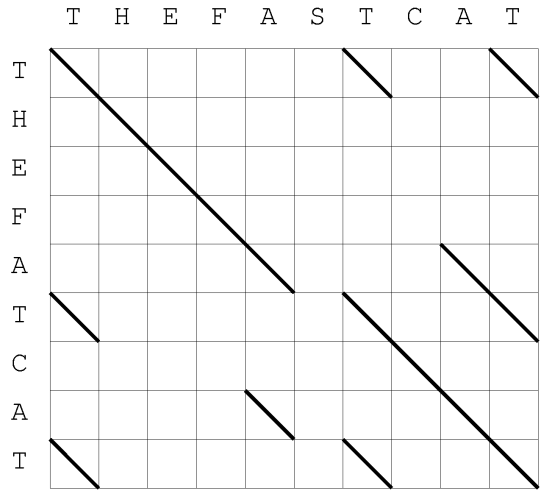

Abbildung 3.3.: Punktdiagramm der Sequenzen aus Abbildung 3.2.

Die Bewertung eines Punktdiagramms bleibt dem Betrachter überlassen. Ähnliche Sequenzen zeigen sich durch stark besetzte Punktdiagramme insbesondere in der Nähe der Diagonalen.

Punktdiagramme können bei langen Sequenzen aus kleinen Alphabeten schnell sehr unübersichtlich werden. Sie lassen sich entrauschen, indem nicht alle einzelnen Sequenzpositionen verglichen werden, sondern nur gefensterte Bereiche der Sequenzen. Diese Methoden können grob nach *Fenster-Methoden* (beispielsweise [ML81]) und *Wort-Methoden* ([WL83]) unterschieden werden (zur Vertiefung vgl. [Han01]).

3.2.2. Substitutionsmatrizen

In Substitutionsmatrizen wird die Ähnlichkeit aller Aminosäurenpaare, die sogenannte *log odd ratio*, notiert. Die einfachste Substitutionsmatrix ist die Identitätsmatrix. Hier wird gleichen Aminosäuren eine 1 zugewiesen, unterschiedlichen eine 0.

Wichtige Substitutionsmatrizen für Proteinalignments sind Blosum-Matrizen (***Blo**ck **Su**bstitution **M**atrix*). Sie wurden 1992 von J. und S. Henikoff eingeführt (s. [HH92]). Basierend auf lokalen Alignments von Sequenzen, die zu einem bestimmten Prozentsatz identisch sind, wird die Substitutionshäufigkeit und damit die relative Austauschwahrscheinlichkeit abgeleitet. Die Blosum62-Matrix benutzt zur Berechnung ausschließlich Sequenzen mit einer Identität von mehr als 62%.

Zur vertiefenden mathematischen und biologischen Motivation von Substitutionsmatrizen sei auf [Dur98], [Rau01] und [Han01] verwiesen.

	a	r	n	d	c	q	e	g	h	i	l	k	m	f	p	s	t	w	y	v
a	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0
r	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3
n	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3
d	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3
c	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1
q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2
e	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2
g	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3
h	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3
i	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3
l	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1
k	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2
m	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1
f	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1
p	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2
s	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2
t	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0
w	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3
y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1
v	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

Abbildung 3.4.: Die Blosum62-Matrix. Die Werte stellen skalierte und logarithmierte, auf ganze Zahlen gerundete, relative Austauschwahrscheinlichkeiten von Aminosäuren dar.

3.2.3. Gaps

Einen weiteren Faktor zur Bewertung von Sequenzähnlichkeiten sind *Gaps*. So können sich im evolutionären Verlauf nicht nur Aminosäuren ändern, sondern es entfallen auch Aminosäuren aus Proteinen (*Deletion*) oder kommen hinzu (*Insertion*). Dieses Phänomen führt zu Lücken im Alignment, den sogenannten *Gaps*.

Im Gesamtszenario eines Alignments können dafür Gapkosten veranschlagt werden. Verbreitet sind *lineare Scores* $\gamma(g) = -gd$. Dabei bezeichnet g die Länge des Gaps und d die veranschlagten Kosten. *Affine Scores* berechnen sich nach $\gamma(g) = -d - (g-1)e$. Hier bezeichnet d die Anfangskosten eines Gaps, das sogenannte *gap-open penalty*, und e die Kosten für jede Gaperweiterung, das *gap-extension penalty*.

3.2.4. Globale Alignments

Globale Alignments versuchen, zwei Sequenzen in ihrer ganzen Länge zu alignieren. Das Ziel ist, eine Zuordnung zu finden, die einen *Gesamtscore* – die Summe der Zuordnungen unter Berücksichtigung der Austauschwahrscheinlichkeiten und der Gapkosten – maximiert.

Die Anzahl der theoretischen Anordnungen zweier Sequenzen der Länge n zueinander ist

$$\left(\begin{array}{c} 2n \\ n \end{array} \right). \tag{3.4}$$

Es ist offensichtlich, dass die optimale Zuordnung nicht durch einfaches Testen aller möglichen Kombinationen lösbar ist. Needleman und Wunsch stellten 1970 einen Algorithmus vor, der die optimale Zuordnung in quadratischer Zeit- und Speicherkomplexität durch ein dynamisches Programm löst (vgl. [NW70]).

Seien $\mathbf{R} = r_1..r_m$ und $\mathbf{S} = s_1..s_n$ zwei univariate Sequenzen der Längen m und n. Der Needleman-Wunsch-Algorithmus erstellt eine Matrix $\mathbf{F}$, in der die Position f_{ij} den maximalen Gesamtscore für die Sequenzen $r_1..r_i$, und $s_1..s_j$ enthält. Damit bezeichnet der Matrixwert f_{mn} den optimalen Gesamtscore der kompletten Sequenzen $\mathbf{R}$ und $\mathbf{S}$.

Der Gesamtscore wird nach folgendem Schema berechnet:

$$f_{0j} = -jc, \tag{3.5}$$

$$f_{i0} = -ic, \tag{3.6}$$

$$f_{ij} = \max \left\{ \begin{array}{l} f_{i-1,j-1} + score(r_i, s_j), \\ f_{i-1,j} - c, \\ f_{i,j-1} - c. \end{array} \right. \tag{3.7}$$

Da für die Berechnung von f_{ij} ausschließlich das Feld unmittelbar rechts ($f_{i-1,j}$), das Feld unmittelbar über ($f_{i,j-1}$) und das Feld diagonal rechts oben ($f_{i-1,j-1}$) betrachtet werden müssen, lässt sich der Gesamtscore f_{mn} elegant rekursiv berechnen. Der $score(r_i, s_j)$ wird im molekularbiologischen Setting durch die Substitutionsmatrizen bestimmt, c sind die Gapkosten.

In einem zweiten Schritt, dem sogenannten *Traceback*, werden die optimalen Zuweisungen ermittelt, indem der zum Gesamtscore korrespondierende *Pfad* durch die Matrix zurück verfolgt wird. Beginnend in f_{mn} werden diejenigen Elemente der Sequenzen einander zugeordnet, die gemäß dem dynamischen Programm aus Gleichungen 3.5 bis 3.7 der aktuellen Position f_{ij} voran gegangen sind.

In der Praxis wird parallel zu $\mathbf{F}$ eine Rückverzeigerungsmatrix $\mathbf{\Phi}$ angelegt, in der der optimale Vorgänger eines jeden f_{ij} gespeichert wird.

3.2.5. Lokale Aligments

Molekurlarbiologen sind häufig an der Erkennung von ähnlichen Teilsequenzen innerhalb längerer Sequenzen interessiert. Ein Algorithmus, der dies leistet, ist der Smith-Waterman-Algorithmus ([SW81a], [SW81b]). Er funktioniert ähnlich dem Needleman-

Wunsch-Algorithmus

$$f_{0j} = 0, \tag{3.8}$$

$$f_{i0} = 0, \tag{3.9}$$

$$f_{ij} = \max \begin{cases} 0, \\ f_{i-1,j-1} + score(s_i, r_j), \\ f_{i-1,j} - c, \\ f_{i,j-1} - c, \end{cases} \tag{3.10}$$

nur dass als zusätzliche Alternative der Wert 0 für jedes f_{ij} hinzugekommen ist. Lokale Alignments sind dann durch Bereiche negativer Scores eingegrenzt. Der maximale Wert f_{ij} der dabei entstehenden Matrix $\mathbf{F}$ zeichnet die Region mit größter lokaler Ähnlichkeit aus. Das *Traceback* funktioniert dann analog zum globalen Fall.

Die beiden hier beschriebenen Algorithmen arbeiten mit *linearen* Gapkosten, lassen sich aber leicht auf *affine* Gapkosten erweitern (vgl. [Dur98]).

3.2.6. Heuristisches Alignment

Während die in den Abschnitten 3.2.4 und 3.2.5 beschriebenen Algorithmen garantiert die optimalen Alignments zweier Sequenzen finden, werden aufgrund der immer noch quadratischen Zeitkomplexität des Needleman-Wunsch- und des Smith-Waterman-Algorithmus für große Protein- und Gendatenbanken heuristische Ansätze benötigt. Im Allgemeinen wird im ersten Schritt eine Indexsuche auf den Sequenzen der Datenbank durchgeführt, um so geeignete Kandidaten für die Weiterverarbeitung zu finden. Für diese wird dann in weiteren Schritten der Score abgeschätzt.

FASTA (s. [PL88]) und BLAST (***B**asic **L**ocal **A**lignment **S**earch **T**ool*, [AL90]) sind die wichtigsten heuristischen Algorithmen, die ein Alignment ermöglichen, darüberhinaus wurden in den letzten Jahren diverse Weiterentwicklungen und Alternativen vorgestellt.

3.3. Multiple Alignments

Ein multiples Sequenz-Alignment (MSA) ist die Ausrichtung *mehrerer* Sequenzen zueinander, wobei die homologen Sequenzpositionen in den Spalten aligniert werden (s. Abb. 3.5).

```
Sequenz 1: KALENDER--
Sequenz 2: CALENDAR--
Sequenz 3: CALENDARIO
Konsens  : CALENDAR--
```

Abbildung 3.5.: Das multiple Alignment der drei Sequenzen zeigt die Verwandschaft der drei Sprachen Deutsch, Englisch und Spanisch.

Zweck ist die Identifikation von Sequenzpositionen, an denen spezifische Aminosäuren für die strukturelle Integrität bzw. die Funktion der Proteine wichtig sind. Dabei sind sogenannte *konservierte Positionen* – Spalten, in denen viele Sequenzen die gleichen Merkmale vorweisen – wichtiger für die Funktion als andere. Weiterhin dienen MSAs dazu, spezifische Signaturen für Proteinfamilien zu erstellen sowie zur Sequenzklassifikation und zur Erstellung phylogenetischer Bäume (s. [Dur98]).

3.3.1. Berechnung multipler Alignments

Hochwertige multiple Alignments werden oft von Experten in Handarbeit erstellt. Da dies ein aufwändiger und teurer Prozess ist, versucht man automatische MSAs ähnlich wie in Abschnitt 3.2 mit Hilfe des Computers zu berechnen.

Das Prinzip der dynamischen Programmierung kann leicht auf mehr als zwei Sequenzen generalisiert werden. Dazu erweitert man die Distanzmatrix mit jeder Sequenz um eine Dimension – für drei zu alignierende Sequenzen ergibt sich somit ein Quader – und passt das dynamische Programm entsprechend an (*multidimensional dynamic programming*, s. [Dur98]). Zwar lässt sich so eine optimale Lösung finden, jedoch beträgt der Rechenaufwand für N Sequenzen mit durchschnittlicher Länge $\bar{T}$ immerhin $O(2^N \bar{T}^N)$, was nur für sehr kleine N in akzeptabler Zeit berechenbar ist.

Daher werden automatische MSAs mittels *progressiver* Verfahren erstellt, wobei sukzessive paarweise Alignments gebildet werden. Ausgangspunkt sind zwei Sequenzen, die ein Alignment ergeben, welches durch Hinzunahme einer dritten Sequenz wieder ein Alignment bildet usw. Das entstehende MSA ist dabei abhängig von der Reihenfolge der verwendeten Sequenzen und lässt somit viel Spielraum für Heuristiken. Es existieren daher auch verschieden Varianten, um progressive MSAs zu bilden (s. [Dur98]).

Das wohl bekannteste dieser Verfahren ist *ClustalW* (s. [HTG94]). ClustalW ist *profilbasiert*, d.h. während der Erstellung des MSA werden erkennbare konservierte Positionen zur Bewertung der folgenden paarweisen Alignments herangezogen (z.B. durch veränderte *mismatch-* oder *gap-penalties*). Nachdem die Distanzmatrix der $\frac{N(N-1)}{2}$ Sequenzpaare in evolutionäre Distanzen konvertiert ist, werden die Alignments durch agglomeratives Clustering nach absteigender Ähnlichkeit zusammengefasst. Dabei kommen verschiedene Heuristiken zum Einsatz. Ein Beispiel für ein mit ClustalW erzeugtes MSA zeigt Abbildung 1.3.

3.3.2. Bewertung eines multiplen Alignments

Im Allgemeinen gibt es immer mehrere korrekte evolutionäre MSAs ([Dur98]). Daher muss die Güte der Alignments untereinander vergleichbar sein. Dies geschieht mittels einer *Scoring-Funktion*, welche üblicherweise die statisitische Unabhängigkeit der einzelnen Spalten annimmt, so dass die Scoring-Funktion eines MSAs m mit I Spalten als

$$S(m) = G + \sum_{i=1}^{I} S(m_i) \tag{3.11}$$

geschrieben werden kann, wobei G eine Scoring-Funktion für die Gaps ist.

Beispielhaft sei hier die *Minimale Entropie* als Scoring-Funktion erwähnt, welche der Spalte m_i gemäß des Entropiemaßes aus der Informationstheorie den Score

$$S(m_i) = -\sum_{a=1}^{A} c_a^i \log p_a^i \tag{3.12}$$

zuweist, wobei p_a^i die Wahrscheinlichkeit eines Sequenzmerkmals $a = 1, .., A$ in Spalte $i = 1, .., I$ ist. Diese kann mittels der Anzahl der auftretenden Merkmale (*counts*) c_a^i gemäß

$$p_a^i = \frac{c_a^i}{\sum_{a'=1}^{A} c_{a'}^i} \tag{3.13}$$

geschätzt werden. Komplett konservierte Spalten erzielen somit einen Score von 0. Die Scores werden in einer Konsensuszeile unterhalb des MSAs zusammengefasst und ggf. entsprechend visualisiert.

3.4. Hidden-Markov-Modelle

HMMs werden in der Bioinformatik dazu verwendet, um paarweise und multiple Alignments zu erstellen. Ein gute Einführung in paarweise Alignments mittels sogenannter *Pair-HMMs* bietet [Dur98], wir beschränken uns im folgenden auf multiple Alignments mit HMMs.

3.4.1. Profil-Hidden-Markov-Modelle

Funktionale biologische Sequenzen treten üblicherweise in Familien (also Mengen mehrerer untereinander ähnlicher Sequenzen) auf. Daher ist es von großer Bedeutung, den Bezug einzelner (neuer) Sequenzen zu den bekannten Sequenzfamilien zu untersuchen.

Um konservierte Merkmale ganzer Familien zu berücksichtigen bietet es sich an, ein Modell jeder Familie zu lernen. Die verbreiteteste Methode, um probabilistische Modelle aus einer Menge gegebener Sequenzen zu bilden, sind die *Profil-Hidden-Markov-Modelle* (PHMM). PHMMs wurden in [Kro93] und [Kro94] eingeführt, um Proteine zu modellieren bzw. um Proteinsequenzen zu klassifizieren. Eine gute Übersicht zu PHMMs in der Bioinformatik bieten [Edd96], [Edd98] und [CTZ04].

Für die Verwendung von PHMMs existieren verschiedene Werkzeuge wie z.B. *HMMER* (s. [SED97]) oder *SAM* (vgl. [SAM95]).

Modellierung von Proteinsequenzen

Zur Bewältigung der oben beschriebenen Aufgaben muss ein HMM mit Rücksicht auf die spezielle Struktur von Proteinen modifiziert werden. Dazu führen wir besondere Zustände und Emissionsmodelle ein.

Die konservierten Bereiche von Proteinsequenzen werden mit Hilfe wiederkehrender *match-Zustände* M_i charakterisiert. Diese emittieren mit der Wahrscheinlichkeit $b_{M_i}(a)$ eine Aminosäure a. Dabei sind nur Zustandsübergänge von M_i zu M_{i+1} möglich. Anfang und Ende der Sequenzen werden durch *stille Zustände* (*silent states*) – Zustände ohne Emissionen – modelliert.

Zur Berücksichtigung von Gaps im Alignment führen wir *insert-Zustände* (Insertionen) I_i ein. Übergänge sind jetzt zusätzlich von M_i zu I_i, von I_i zu I_i (*loop*) und von I_i zu M_{i+1} möglich.

In ähnlicher Weise werden für Sprünge im Alignment durch *deletion-Zustände* nachgebildet, wobei diese still (also nichtemittierend) sind. Damit ergeben sich weitere Möglichkeiten für Zustandsübergänge, welche im Beipielmodell in Abbildung 3.6 dargestellt sind.

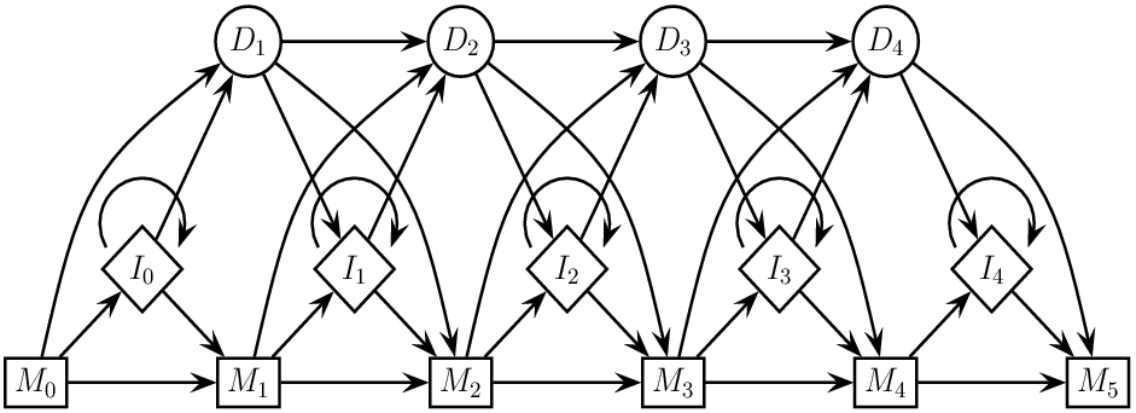

Abbildung 3.6.: Zustandsübergangsstruktur eines PHMMs mit den Anfangs- und Endzuständen M_0 bzw. M_5. Zur Verdeutlichung sind die unterschiedlichen Zustandsarten durch verschieden Symbole dargestellt (aus [Edd96]).

Initialisierung und Parameterschätzung

Damit die positionsspezifische Information des multiplen Alignments repräsentiert werden kann, müssen die Zustandsübergangs- und Emissionswahrscheinlichkeiten aus den gegebenen Sequenzen geschätzt werden. Dafür ist bei nahezu allen PHMMs eine Initialisierung mittels eines vorangestellten oder bestehenden MSAs (z.B. durch ClustalW) notwendig. Dies stellt eine starke Einschränkung der Verwendbarkeit ein, da eine schlechte Initialisierung das Ergebnis stark verfälschen kann und die Erstellung eines MSAs rechenaufwändig bzw. teuer ist (s. Abschnitt 3.3). Zu beachten ist außerdem die Wahl der Modellgröße, also die Anzahl der match-Zustände.

Bei der Parameterschätzung (s. Abschnitt 2.5) kann entweder das Baum-Welch oder das Viterbi-Training verwendet werden. Zur Schätzung der Wahrscheinlichkeiten werden wiederum die beobachteten Auftrittshäufigkeiten der Zustandsübergänge und Emissionen der Sequenzmerkmale gezählt und zur Gesamtzahl ins Verhältnis gesetzt.

Klassifikation neuer Sequenzen

Mit dem fertigen Modell können Sequenzen nach ihrer Produktionswahrscheinlichkeit $P(\mathbf{S}|\Lambda)$ bzw. $P(\mathbf{S}, \mathbf{q}^*|\Lambda)$ bewerten werden. Ersteres kann mittels Vorwärts-, letzteres durch den Viterbi-Algorithmus berechnet werden. Für M Klassen sind dementsprechend M Modelle zu trainieren, die Klassifikation erfolgt gemäß

$$f(\mathbf{S}) = \arg\max_{y_i} P(\mathbf{S}|\Lambda_{y_i}) \tag{3.14}$$

bzw.

$$f(\mathbf{S}) = \arg\max_{y_i} P(\mathbf{S}, \mathbf{q}^*|\Lambda_{y_i}). \tag{3.15}$$

3.5. Kernmethoden

Das Prinzip der Ähnlichkeits- bzw. Skalarproduktkerne (s. Abschnitt 2.7.4) wird in der Bioinformatik genutzt, um Homologien von Proteinen zu untersuchen (z.B. [Sai04], [Bus04], [Lai02], [Jaa99], [Jaa98]), Proteinklassifikation zu betreiben (z.B. [Les04]) oder sogenannte *Promoterregionen* (Chromosombereiche, welche Transskriptionsbedingungen für Gene enthalten) zu identifizieren (z.B. [Gor03]).

Dabei gibt es mehrere Möglichkeiten die Ähnlichkeit der (Sub-)Sequenzen zu messen. Eine besteht darin, eine geeignete Vektorrepräsentation für die Sequenzen zu wählen und dann das Skalarprodukt auf diesen zu berechnen (z.B. [Jaa99]). Eine andere geht von einem Ähnlichkeitsmaß aus, welches als relevant für die Zielanwendung bekannt ist und formt dieses in eine Kernfunktion um (z.B. [Sai04]).

In der Praxis werden meist Support-Vektor-Maschinen (s. Abschnitt 2.7) verwendet, da diese zum diskriminativen Lernen im *Feature Space* gut geeignet sind und zahlreiche Toolboxen existieren.

Die Kernmethoden haben sich als sehr erfolgreich in den oben erwähnten Anwendungen herausgestellt, jedoch beinhalten sie auch einige Probleme. So kann bei einigen der als diskriminativ beworbenen Verfahren festgestellt werden, dass nur der 2. Schritt wirklich diskriminativ ist, der vorausgehende und unabhängige Vektorisierungsschritt dagegen nicht.

Als noch wichtiger einzustufen ist der immense Zeit- und Speicherplatzbedarf der Verfahren. Für die Berechnung der Kernmatrix sind N^2 Kernfunktionsauswertungen erforderlich, was für realistische Anwendungen mit größeren N problematisch ist.

4. Die neuen Ansätze im Überblick

4.1. Grundlegende Eigenschaften der Verfahren

Ordered-Means-Modelle (OMMs) sind endliche Automaten mit K Zuständen, wobei jeder Zustand durch einen Prototypvektor $\boldsymbol{\mu}_k$ repräsentiert wird. Eine einfache theoretische Herangehensweise ist es, sich ein HMM ohne explizite Übergangswahrscheinlichkeiten vorzustellen.

Feature-Alignment-Maschinen (FAMs) dagegen realisieren eine adaptive und iterativ optimierte Merkmalsselektion. Die Hauptleistung des Verfahrens liegt dabei in der Transformation der Sequenzen in einen einheitlichen Vektorraum, in welchem gängige Methoden des ML direkt anwendbar sind.

4.2. Notation

In den folgenden Kapiteln gelten die folgenden Konventionen. Eine *Sequenz* wird durch $\mathbf{S}$ ausgedrückt. Sie ist eine Matrix von *Sequenzvektoren* $\mathbf{s}_t$, wobei t den t-ten Spaltenvektor der Sequenz kennzeichnet. Sowohl die OMMs als auch die FAMs arbeiten prototypbasiert (bis auf den *OMMKDE*) mit den entsprechenden *Prototypen* $\mathbf{W}$. Eine Zuordnung einer Sequenz zu einem Prototypen ist durch die Zuordnungsmatrix $\mathbf{Z}$ gegeben.

Im Umfeld der OMMs werden die einzelnen Prototypvektoren in Anlehnung an HMMs $\boldsymbol{\mu}_k$ genannt.

Die FAMs arbeiten mit sogenannten *Sequenzmusterfolgen* $\bar{\mathbf{S}}$, die aus einzelnen *Sequenzmustervektoren* $\bar{\mathbf{s}}_t$ bestehen. Die für die Idee der *FAMs* wichtigen Merkmalsraumprojektionen der Sequenzen werden durch $\mathbf{x}$ beschrieben, die Merkmalsraumprojektionen der Prototypen durch $\mathbf{w}$.

Eine detaillierte Liste zur Notation ist in Anhang B zu finden.

4.3. Die prinzipiellen Unterschiede im Alignment

4.3.1. Ordered-Means-Modelle

Das Alignmentprinzip der Ordered-Means-Modelle lautet: „Jeder Sequenzvektor wird genau einem Referenzvektor zugeordnet, wobei Mehrfachzuordnungen möglich sind“. D.h. ein Referenzvektor kann mehrere Sequenzvektoren einer Sequenz auf sich vereinen. Abbildung 4.1 veranschaulicht das Alignmentprinzip der probabilistischen Variante *OMMall* in logarithmischer Repräsentation (s. Abschnitt A.1).

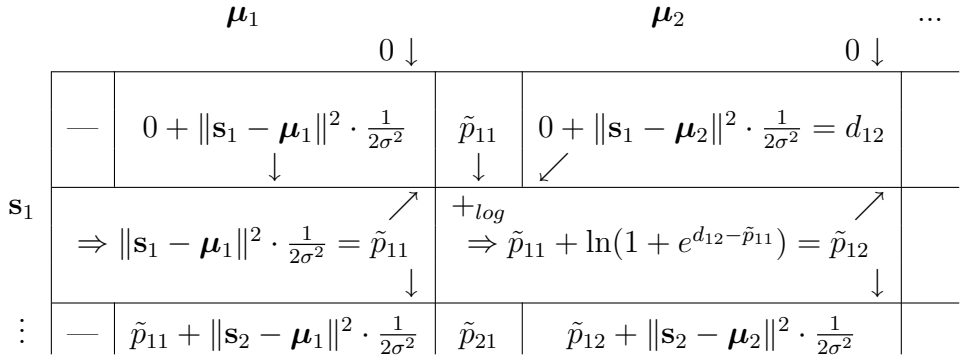

Abbildung 4.1.: Funktionsweise des dynamischen Programms im Alignmentschritt von *OMMall*. Die Sequenzvektoren $\mathbf{s}_1..\mathbf{s}_T$ und die Prototypvektoren $\boldsymbol{\mu}_1..\boldsymbol{\mu}_K$ werden an den Rändern einer Matrix repräsentiert, um die Werte in den Zellen zu berechnen. Jede Zelle ist in drei Felder aufgeteilt, in denen in Abhängigkeit von der Sequenz- (t) und Zustandsposition (k) der Übertrag aus dem letzten Zustand (linkes oberes Feld), die Summe aus Übertrag von letztem Zeitschritt und aktueller σ-gewichteter Distanz von Sequenzvektor $\mathbf{s}_t$ zu Prototypvektor $\boldsymbol{\mu}_k$ (rechtes oberes Feld) und die Logarithmussumme beider Felder (unteres Feld), welche das Zellenendergebnis darstellt. Die Pfeile veranschaulichen die Übertragsrichtung.

Für $\sigma \to 0$ geht der Algorithmus – durch die Minimumentscheidung und die Dominanz der Distanzterme – in den deterministischen Fall *OMMbest* und damit in den Viterbi-Algorithmus über (s. Abb. 4.2).

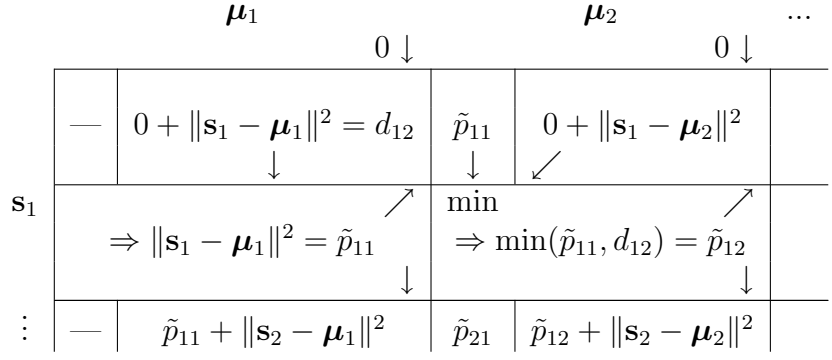

Abbildung 4.2.: Funktionsweise des dynamischen Programms im Alignmentschritt von *OMMbest*. Im Unterschied zu Abbildung 4.1 werden die Distanzen unverändert verwendet und statt der logarithmischen Summe das Minimum der beiden oberen Felder als Zellenergebnis berechnet.

Anhand einer Beispielsequenz und einer Beispielreferenzsequenz stellen wir die kumulative Distanzmatrix und die Rückverzeigerung (*Traceback*) für *OMMbest* (deterministi-

sche Zuordnungen) dar:

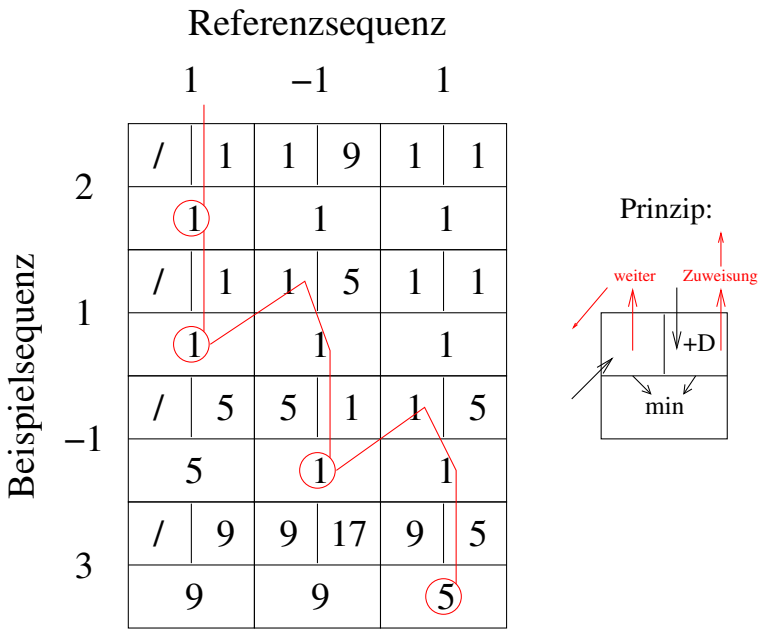

Abbildung 4.3.: Veranschaulichung des Alignmentprinzips der *Ordered-Means-Modelle* anhand eines Beispiels. In die Felder haben wir die Werte der an den Rändern abgebildeten eindimensionalen Beispielsequenzen eingesetzt. Ausgehend vom rechten unteren Feld findet die Rückverzeigerung nach dem rechts abgebildeten Prinzip (rote Pfeile) statt und ergibt den roten Pfad. Die zugeordneten Sequenzvektoren sind durch rote Kreise gekennzeichnet.

In dem Beipiel in Abbildung 4.3 werden also die ersten beiden Sequenzvektoren dem ersten Referenzvektor, der dritte Sequenzvektor dem zweiten und der vierte Sequenzvektor dem dritten Referenzvektor zugeordnet.

4.3.2. Feature-Alignment-Maschinen

Bei den Feature-Alignment-Maschinen lautet das Alignmentprinzip: „Genau ein Sequenzvektor wird jedem Referenzvektor zugeordnet, wobei Mehrfachzuordnungen nicht erlaubt sind“ (vgl. Abb. 4.4). Dies bedeutet, dass im Allgemeinen nicht alle Sequenzvektoren zugeordnet werden und somit nicht in den nachfolgenden Berechnungsschritt einfließen. Mit dieser bijektiven Zuordnung ist sichergestellt, dass die transformierten Sequenzen im selben Merkmalsraum liegen.

Mit den selben Sequenzen wie im vorigen Unterabschnitt sieht die kumulative Score-Matrix und die daraus resultierende Rückverzeigerung also aus wie in Abbildung 4.5. Der durch das Alignment entstehende Merkmalsvektor lautet dann $\mathbf{x} = [2, 3, 6]^T$.

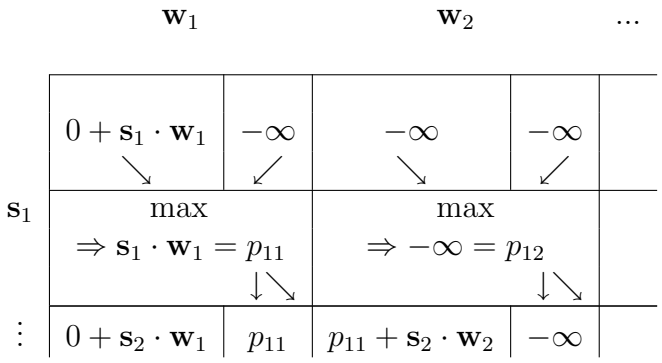

Abbildung 4.4.: Funktionsweise des Alignmentprinzips der *Feature-Alignment-Maschinen* als dynamisches Programm. Wieder ist die Aufteilung der Zellen in drei Felder zu erkennen, wobei das linke obere Feld jetzt eine Summe aus dem aktuellen Score und dem Zelleninhalt des vorigen Sequenz- und Prototypvektors ist. Das rechte obere Feld beinhaltet den Score-Übertrag aus dem letzten Zeitschritt, das untere Feld das Maximum der beiden oberen Felder.

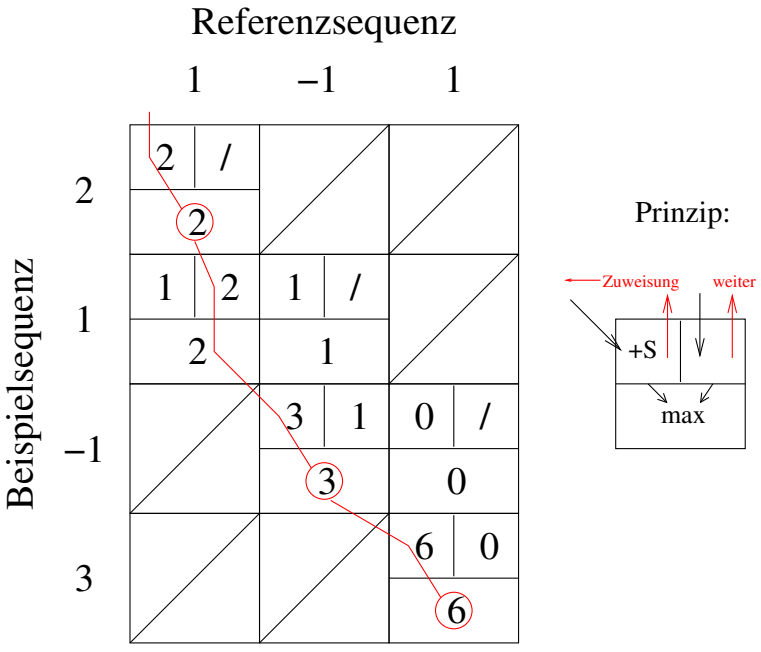

Abbildung 4.5.: Veranschaulichung des Alignmentprinzips der *Feature-Alignment-Maschinen* anhand eines Beispiels. Zu den Erklärungen aus Abbildung 4.3 ist hinzuzufügen, dass gemäß dem Alignmentprinzip unerlaubte Zuordnungen durch einen Schrägstrich (keine Auswertung) gekennzeichnet sind.

4.4. Taxonomie

In Tabelle 4.1 sind einige Eigenschaften der Algorithmen überblickhaft dargestellt. Im einzelnen bedeuten diese:

prototypbasiert: Repräsentiert das gelernte Modell alle Trainingsdaten der Klasse? Ein echtes multiples Alignment setzt ein prototypbasiertes Verfahren voraus.

generativ: Wird ein Modell gelernt, dass auch zum *Generieren* von Sequenzen benutzt werden kann?

diskriminativ: Repräsentiert das gelernte Modell die *Unterschiede* mehrerer Klassen oder ausschließlich eine Klasse?

Methode	prototypbasiert	generativ	diskriminativ
OMMbest	Ja	Ja	Nein
OMMall	Ja	Ja	Nein
OMM KDE	Nein	Ja	Nein
FAM Mean	Ja	Nein	Nein
FASVM	Ja	Nein	Nein
FASVM diskriminativ	Ja	Nein	Ja

Tabelle 4.1.: Gegenüberstellung einiger Eigenschaften der Algorithmen.

5. Ordered-Means-Modelle

5.1. Idee und Motivation

Ordered-Means-Modelle (OMMs) sind theoretisch verwandt mit den Hidden-Markov-Modellen (s. Abschnitt 2.5). Die Idee hinter den OMMs besteht darin, keine Zustandsübergänge durch Übergangswahrscheinlichkeiten zu bevorzugen. Vielmehr sind alle Pfade gleichwahrscheinlich, wobei jedoch keine Übergänge zu vorherigen Zuständen erlaubt sind (lineares Modell).

Der Name erklärt sich aufgrund dieser ordnungserhaltenden Abbildung der Sequenzvektoren auf die Referenzvektoren, die durch Mittelwerte (*means*) der Emissionsdichten repräsentiert werden. Das Alignment-Prinzip lautet in diesem Fall: „Jeder Sequenzvektor wird genau einem Referenzvektor zugeordnet“, womit Zuordnungen mehrerer Sequenzvektoren auf einen Referenzvektor möglich sind.

Ziel ist das Lernen eines generativen Modells (generierender endlicher Automat), welches für sich genommen leicht interpretierbar und als Klassifikator (bzw. als Grundlage für andere Algorithmen des maschinellen Lernens) verwendbar ist.

Im Folgenden geben wir als erstes die Spezifikation der OMMs an und beschreiben das Problem der kombinatorischen Explosion der Anzahl der Pfadmöglichkeiten (Abschnitt 5.2). Danach besprechen wir die beiden prototypbasierten Modellvarianten *OMMall* (Abschnitt 5.3) und *OMMbest* (Abschnitt 5.4) und erläutern die Verwendung als Klassifikator (Abschnitt 5.5). Als letztes beschreiben wir den Spezialfall des paarweisen Alignments mit OMMs, welcher zur Kerndichteschätzung herangezogen werden kann (Abschnitt 5.6).

5.2. Spezifikation

Ein OMM Ω benötigt im Gegensatz zu einem HMM keine Anfangs- und Übergangswahrscheinlichkeiten. Es wird allein durch die Größe des Modells K und die Emissionsdichten

$$b_k(\mathbf{s}_t) = \mathcal{N}(\mathbf{s}_t; \boldsymbol{\mu}_k, \sigma) = \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} \cdot e^{-\frac{1}{2\sigma^2} \cdot \|\mathbf{s}_t - \boldsymbol{\mu}_k\|^2} \tag{5.1}$$

spezifiziert. Dabei ist $k = 1, .., K$ ein *Zustand* (eine latente Variable), $\boldsymbol{\mu}_k \in \mathbb{R}^d$ der Referenz- bzw. Mittelwertvektor zum Zustand k, $t = 1, .., T$ ein Zeitpunkt der Sequenz $\mathbf{S}$ mit dem zugehörigen Merkmalsvektor $\mathbf{s}_t \in \mathbb{R}^d$.

Der Parameter σ gibt die Standardabweichung der Daten vom Mittelwert der Verteilung an. Ein großes σ bedeutet, dass die Daten nicht sehr stark um den Mittelpunkt

konzentriert sind und somit eine flache Verteilung mit weitreichenden Ausläufern (*tails*) modelliert wird. Der Distanzterm im Exponenten wird dann sehr klein, d.h. selbst große Distanzen werden noch als lokal gewertet, da die Dichte noch relativ groß ist. Ein kleines σ dagegen sorgt für einen großen Anteil des Distanzterms, also für starke Lokalität und Konzentration um den Mittelwert. Für große Distanzen sind die Dichten somit sehr klein. Über die Größe von σ lässt sich also der Einfluss entfernter Sequenzvektoren auf die Emissionsdichte bzgl. des Referenzvektors steuern.

Die Pfadproduktdichte – die Wahrscheinlichkeitsdichte, eine Sequenz $\mathbf{S}$ zu beobachten, wenn der Pfad $\mathbf{q} = q_1..q_T$, also die Belegung der latenten Variablen, und das Modell Ω gegeben sind – ist

$$\begin{aligned} p(\mathbf{S}|\mathbf{q},\Omega) &= \prod_{t=1}^{T} p(\mathbf{s}_t|q_t,\Omega) \\ &= \prod_{t=1}^{T} b_{q_t}(\mathbf{s}_t). \end{aligned} \tag{5.2}$$

Hierbei wird statistische Unabhängigkeit der Observationen angenommen, d.h. es gilt

$$p(b_{q_1}(\mathbf{s}_1),..,b_{q_T}(\mathbf{s}_T)) = p(b_{q_1}(\mathbf{s}_1)) \cdot .. \cdot p(b_{q_T}(\mathbf{s}_T)).$$

Die Wahrscheinlichkeit eines Pfades $\mathbf{q}$ aus der Menge der Pfade Q mit $|Q| = M$ ist unabhängig vom Pfad und beträgt bei gegebenem Modell Ω und Sequenz $\mathbf{S} = \mathbf{s}_1..\mathbf{s}_T$

$$\begin{aligned} P(\mathbf{q}|\Omega) &= \frac{1}{M} = \frac{1}{\#\text{ möglicher Pfade durch das Modell}} \\ &= \frac{1}{\binom{K+T-1}{T}} = \frac{1}{\frac{(K+T-1)!}{(K-1)!\cdot T!}} = \frac{(K-1)!\cdot T!}{(K+T-1)!}. \end{aligned} \tag{5.3}$$

Die Anzahl der Pfade ist hier dadurch beschränkt, dass nur *lineare Modelle* – d.h. Modelle, bei denen die Zustände nur aufsteigend oder gleichbleibend durchlaufen werden können – verwendet werden. In der Kombinatorik wird diese Konstellation als *Kombination mit Wiederholung* (Kombination heißt: ohne Berücksichtigung der Reihenfolge) bezeichnet. Der Ausdruck $M = \frac{(K+T-1)!}{(K-1)!\cdot T!}$ ergibt sich aus der Anzahl der Möglichkeiten, T Objekte aus einer Menge von K Objekten auszuwählen. Da die Reihenfolge keine Rolle spielt, können die Elemente als monoton steigend sortiert notiert werden. Insgesamt ergeben sich $T + K + 1$ Positionen, aus denen T beliebig gewählt werden können, also $\binom{K+T-1}{T}$ Möglichkeiten. Um den explosionsartigen Anstieg der Anzahl der Pfade mit steigender Anzahl der Zustände (K) und Zeitpunkte (T) zu verdeutlichen, haben wir in Tabelle 5.1 einige kleine und zwei realistische Beispiele aufgezählt. Diese Beispiele verdeutlichen, dass sich die explizite Berechnung der Pfadproduktdichten für alle Pfade verbietet.

Anwendung	K	T	M
-	3	4	15
-	5	10	1001
-	20	10	20030010
EEG-Datenanalyse	20	100	$\approx 4.91 * 10^{21}$
Proteinklassifikation	200	400	$\approx 8.35 * 10^{163}$

Tabelle 5.1.: Anzahl der möglichen Pfade M für verschiedene Modellgrößen K und Sequenzlängen T.

5.3. OMMall

Bei der Variante *OMMall* berücksichtigen wir alle Pfade durch das Modell. Die Zuordnungen der Sequenzvektoren auf die Referenzvektoren sind probabilistisch, d.h. jeder Merkmalsvektor $\mathbf{s}_t$ einer Sequenz geht aufgrund von Wahrscheinlichkeiten in die Prototypenberechnung ein. Dies realisieren wir durch kontinuierliche Verantwortlichkeiten (*Responsibilities*) $r_{kt} \in [0, 1]$ je Sequenz, wobei für jeden Zeitpunkt der Sequenz $t = 1, .., T$

$$\sum_{k=1}^{K} r_{kt} = 1 \tag{5.4}$$

gelten muss.

Die Verbundwahrscheinlichkeitsdichte, die Sequenz $\mathbf{S}$ und den Pfad $\mathbf{q}$ bei gegebenem Ω zu beobachten ist

$$\begin{aligned} p(\mathbf{S}, \mathbf{q}|\Omega) &= p(\mathbf{S}|\mathbf{q}, \Omega) \cdot P(\mathbf{q}|\Omega) && (5.5) \\ &= \frac{1}{M} \prod_{t=1}^{T} b_{q_t}(\mathbf{s}_t). && (5.6) \end{aligned}$$

Die Produktionswahrscheinlichkeit ist dann die Summe aller möglichen Pfade in Gleichung 5.5:

$$\begin{aligned} P(\mathbf{S}|\Omega) &= \sum_{\mathbf{q} \in Q} p(\mathbf{S}, \mathbf{q}|\Omega) \\ &= \frac{1}{M} \sum_{\mathbf{q} \in Q} \prod_{t=1}^{T} b_{q_t}(\mathbf{s}_t). && (5.7) \end{aligned}$$

Zielfunktion und Parameterschätzung

Die Schätzung der einzigen Modellparameter – der Mittelwerte $\boldsymbol{\mu}_k$ – findet durch eine *Maximum-Likelihood-Schätzung* statt. Dabei wird jedoch nicht die Likelihood $\mathcal{L}_N = \prod_{i=1}^{N} P(\mathbf{S}^i|\Omega)$, sondern die log-Likelihood $L_N = \ln(\prod_{i=1}^{N} P(\mathbf{S}^i|\Omega))$ der Daten (also der

Sequenzen $\{\mathbf{S}^1, .., \mathbf{S}^N\}$) maximiert:

$$\begin{aligned}
\max_{\Omega} L_N(\Omega) &= \max_{\Omega} \sum_{i=1}^{N} \ln P(\mathbf{S}^i|\Omega) \\
&= \max_{\Omega} \sum_{i=1}^{N} \ln \sum_{\mathbf{q}\in Q^i} p(\mathbf{S}^i|\mathbf{q},\Omega) \cdot P(\mathbf{q}|\Omega) \qquad (5.8) \\
&= \max_{\Omega} \sum_{i=1}^{N} \ln \left(\frac{1}{M^i} \sum_{\mathbf{q}\in Q^i} \prod_{t=1}^{T^i} b_{q_t}(\mathbf{s}_t^i) \right). \qquad (5.9)
\end{aligned}$$

Die direkte Maximierung ist (aufgrund des Logarithmus vor der Summation) für unsere Zufallsvariablen nicht einfach möglich, jedoch bietet der EM-Algorithmus (*Expectation-Maximization*, s. [DLR77]) einen Ausweg. Dazu ist es hilfreich, das Modell als Mischung von M^i Mischungskomponenten (*mixture*, in diesem Fall die Pfadproduktdichten $p(\mathbf{S}^i|\mathbf{q},\Omega)$) mit konstanten Mischungsgewichten $\omega_m = \frac{1}{M^i}$ zu betrachten. Die Maximierung der "complete-data-log-likelihood" (der unteren Grenze für L_N) erfolgt mittels Iteration zweier Schritte:

E-Schritt:

$$\begin{aligned}
h_{\mathbf{q}}^i &= P(\mathbf{q}|\mathbf{S}^i,\Omega) \\
&= \frac{p(\mathbf{S}^i|\mathbf{q},\Omega)P(\mathbf{q}|\Omega)}{\sum_{\mathbf{q}'\in Q^i} p(\mathbf{S}^i|\mathbf{q}',\Omega)P(\mathbf{q}'|\Omega)} \overset{(5.3)}{=} \frac{p(\mathbf{S}^i|\mathbf{q},\Omega)}{\sum_{\mathbf{q}'\in Q^i} p(\mathbf{S}^i|\mathbf{q}',\Omega)} \\
&= \frac{\prod_{t=1}^{T^i} b_{q_t}(\mathbf{s}_t)}{\sum_{\mathbf{q}'\in Q^i}\prod_{t=1}^{T^i} b_{q_t'}(\mathbf{s}_t)} = \frac{\left(\frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}}\right)^{T^i} \prod_{t=1}^{T^i} \tilde{b}_{q_t}(\mathbf{s}_t)}{\left(\frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}}\right)^{T^i} \sum_{\mathbf{q}'\in Q^i}\prod_{t=1}^{T^i} \tilde{b}_{q_t'}(\mathbf{s}_t)} \qquad (5.10)
\end{aligned}$$

M-Schritt:

$$\begin{aligned}
\hat{\mathbf{W}} = [\hat{\boldsymbol{\mu}}_1, .., \hat{\boldsymbol{\mu}}_K] &= \arg\max_{\mathbf{W}} \sum_{i=1}^{N} \sum_{\mathbf{q}\in Q^i} h_{\mathbf{q}}^i \ln p(\mathbf{S}^i|\mathbf{q},\Omega) \\
&= \arg\max_{\mathbf{W}} \sum_{i=1}^{N} \sum_{\mathbf{q}\in Q^i} h_{\mathbf{q}}^i \left(\sum_{t=1}^{T^i} \frac{-\|\mathbf{s}_t^i - \boldsymbol{\mu}_{q_t}\|^2}{2\sigma^2} - \frac{T^i d}{2} \ln(2\pi\sigma^2) \right) \\
&= \arg\min_{\mathbf{W}} \sum_{i=1}^{N} \sum_{\mathbf{q}\in Q^i} h_{\mathbf{q}}^i \sum_{t=1}^{T^i} \|\mathbf{s}_t^i - \boldsymbol{\mu}_{q_t}\|^2. \qquad (5.11)
\end{aligned}$$

Die Terme $\frac{T^i d}{2}\ln(2\pi\sigma^2)$ und $2\sigma^2$ im M-Schritt sind bezüglich der Maximierung irrelevant, da sie nicht von $\mathbf{W}$ abhängen. Aus der Normierung im E-Schritt ergibt sich außerdem

$\sum_{\mathbf{q}\in Q^i} h^i_{\mathbf{q}} = 1$. Das Iterationsschema wird solange wiederholt, bis sich der Wert der Zielfunktion (Gl. 5.8) in hinreichender Nähe eines lokalen Maximums befindet.

Berechnung der Verantwortlichkeiten

Die Verantwortlichkeit r^i_{kt} eines Sequenzvektors $\mathbf{s}^i_t$ einer Sequenz $\mathbf{S}^i$ für die Neuschätzung eines Referenzvektors $\boldsymbol{\mu}_k$ ist gegeben durch die Summe der Pfadwahrscheinlichkeiten $h^i_{\mathbf{q}}$ der Pfade, die eine Zuordnung von $\mathbf{s}^i_t$ auf $\boldsymbol{\mu}_k$ vorsehen:

$$\begin{aligned} r^i_{kt} &\equiv \frac{P(\mathbf{S}^i, q_t = k|\Omega)}{P(\mathbf{S}^i|\Omega)} \\ &= \sum_{\substack{\mathbf{q}\in Q:\\ q_t=k}} h^i_{\mathbf{q}} = \frac{\sum\limits_{\substack{\mathbf{q}\in Q:\\ q_t=k}} \prod\limits_{t=1}^{T^i} b_{q_t}(\mathbf{s}^i_t)}{\sum\limits_{\mathbf{q}'\in Q^i} \prod\limits_{t=1}^{T^i} b_{q'_t}(\mathbf{s}^i_t)} \qquad (5.12) \\ \Rightarrow \sum_{k=1}^{K} r^i_{kt} &= \sum_{k=1}^{K} \sum_{\substack{\mathbf{q}\in Q^i:\\ q_t=k}} h^i_{\mathbf{q}} \stackrel{!}{=} 1. \qquad (5.13) \end{aligned}$$

Die Verantwortlichkeiten r^i_{kt} können dann im M-Schritt benutzt werden, um die neuen Prototypen $\mathbf{W} = [\boldsymbol{\mu}_1, .., \boldsymbol{\mu}_K]$ anhand der angepassten Regressionsfunktion

$$\mathbf{W} = \arg\min_{\mathbf{W}} E_{Reg} = \sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{t=1}^{T^i} r^i_{kt} \|\mathbf{s}^i_t - \boldsymbol{\mu}_k\|^2$$

aus den Sequenzvektoren zu schätzen:

$$\begin{aligned} \nabla_{\boldsymbol{\mu}_k} E_{Reg} &= \sum_{i=1}^{N}\sum_{t=1}^{T^i} r^i_{kt} \left(-2\mathbf{s}^i_t + 2\boldsymbol{\mu}_k\right) \stackrel{!}{=} \mathbf{0} \\ \Rightarrow \boldsymbol{\mu}_k &= \frac{\sum\limits_{i=1}^{N}\sum\limits_{t=1}^{T^i} r^i_{kt} \cdot \mathbf{s}^i_t}{\sum\limits_{i=1}^{N}\sum\limits_{t=1}^{T^i} r^i_{kt}}. \qquad (5.14) \end{aligned}$$

5.3.1. Der DP-Matrix-Trick

Setzt man die Kosten für die Berechnung der Pfadproduktdichte eines Pfades mit c an, so benötigt die explizite Berechnung der Produktdichten aller M Pfade $O\left(c \cdot \frac{(K+T-1)!}{(K-1)!\cdot T!}\right)$ Rechenoperationen und ist daher für größere Modelle und Sequenzen nicht mehr durchzuführen. Jedoch erlaubt der Verzicht auf Übergangswahrscheinlichkeiten und das ordnungserhaltende Alignmentprinzip die zusammenfassende Berechnung aller Pfadwahr-

scheinlichkeiten mittels eines dynamischen Programms (DP). Die dadurch als Matrixschema (s. Abb. 4.1) notierbare kumulative Pfadproduktdichtenberechnung ist dabei ebenfalls für die Bestimmung der Verantwortlichkeiten verwendbar. Der Trick besteht also darin, die Produktdichten der möglichen Pfade und die Verantwortlichkeiten implizit zu berechnen.

Der Vorwärtsalgorithmus (FA)

Für allgemeine HMMs lässt sich die Produktionswahrscheinlichkeit $P(\mathbf{S}|\Omega)$ in $O(K^2T)$ Operationen durch den Vorwärtsalgorithmus berechnen (vgl. [Rab89]). Durch den Verzicht auf Übergangswahrscheinlichkeiten stellt sich der Vorwärtsalgorithmus mit der Vorwärtsvariable

$$\alpha_{kt} = p(\mathbf{s}_1..\mathbf{s}_t, q_t = k|\Omega)$$

(die Wahrscheinlichkeitsdichte, die Sequenz bis zum Zeitpunkt t beobachtet zu haben und sich bei t im Zustand k zu befinden) und der Rückwärtsvariable

$$\beta_{kt} = p(\mathbf{s}_{t+1}..\mathbf{s}_T|q_t = k, \Omega)$$

(die Wahrscheinlichkeitsdichte, die Sequenz ab dem Zeitpunkt $t+1$ beobachten zu werden und bei t im Zustand k gewesen zu sein) für OMMs folgendermaßen dar:

$$\begin{aligned} P(\mathbf{S}|\Omega) &= \frac{1}{M}\sum_{\mathbf{q}\in Q}\prod_{t=1}^{T} b_{q_t}(\mathbf{s}_t) && (5.15)\\ &= \frac{1}{M}\sum_{k=1}^{K}\alpha_{kT} && (5.16)\\ \text{mit } \alpha_{kt} &= \left(\sum_{k'=1}^{K}\alpha_{k',t-1}\right) b_k(\mathbf{s}_t) && (5.17)\\ \text{und } \alpha_{k1} &= b_k(\mathbf{s}_t). && (5.18) \end{aligned}$$

Der schnelle Vorwärtsalgorithmus (FFA)

Zusätzlich lässt sich die Komplexität der Aufgabe in diesem speziellen Setting aber weiter auf $O(KT)$ Rechenoperationen reduzieren. Da wir keine rückwärtsgerichteten Zustandsübergänge zulassen, ergibt sich die Produktionswahrscheinlichkeit $P(\mathbf{S}|\Omega)$ zu

$$\begin{aligned} P(\mathbf{S}|\Omega) &= \frac{1}{M}\tilde{\alpha}_{KT} && (5.19)\\ \text{mit } \tilde{\alpha}_{kt} &= \tilde{\alpha}_{k,t-1}b_k(\mathbf{s}_t) + \tilde{\alpha}_{k-1,t}\ , && (5.20)\\ \tilde{\alpha}_{k0} &= 1 && (5.21)\\ \text{und } \tilde{\alpha}_{0t} &= 0. && (5.22) \end{aligned}$$

Dabei bezeichnet

$$\begin{aligned}
\tilde{\alpha}_{kt} &= P(\mathbf{s}_1..\mathbf{s}_t|q_t \leq k, \Omega) && (5.23)\\
&= \sum_{\substack{\mathbf{q}\in Q:\\ q_t\leq k}} p(\mathbf{s}_1..\mathbf{s}_t|\mathbf{q},\Omega) && (5.24)\\
&= \underbrace{\sum_{\substack{\mathbf{q}\in Q:\\ q_t= k}} p(\mathbf{s}_1..\mathbf{s}_t|\mathbf{q},\Omega)}_{\tilde{\alpha}_{k,t-1}b_k(\mathbf{s}_t)} + \underbrace{\sum_{\substack{\mathbf{q}\in Q:\\ q_t< k}} p(\mathbf{s}_1..\mathbf{s}_t|\mathbf{q},\Omega)}_{\tilde{\alpha}_{k-1,t}} && (5.25)
\end{aligned}$$

die Summe aller Produktdichten aller Pfade, die im Zustand $q_t = k$ oder $q_t < k$ den Vektor $\mathbf{s}_t$ emittieren, und zwar vom ersten Sequenzvektor $\mathbf{s}_1$ bis zum t-ten $\mathbf{s}_t$.

Der schnelle Rückwärtsalgorithmus (FBA)

So wie der Vorwärtsalgorithmus lässt sich auch der Rückwärtsalgorithmus von OMMs auf $O(KT)$ Rechenschritte beschleunigen:

$$\begin{aligned}
P(\mathbf{S}|\Omega) &= \frac{1}{M}\tilde{\beta}_{1,1} && (5.26)\\
\text{mit } \tilde{\beta}_{kt} &= \tilde{\beta}_{k,t+1}b_k(\mathbf{s}_t) + \tilde{\beta}_{k+1,t}, && (5.27)\\
\tilde{\beta}_{K+1,t} &= 0 && (5.28)\\
\text{und } \tilde{\beta}_{k,T+1} &= 1. && (5.29)
\end{aligned}$$

Auch hier gilt:

$$\begin{aligned}
\tilde{\beta}_{kt} &= \sum_{\substack{\mathbf{q}\in Q:\\ q_t\geq k}} p(\mathbf{s}_t..\mathbf{s}_T|\mathbf{q},\Omega) && (5.30)\\
&= \underbrace{\sum_{\substack{\mathbf{q}\in Q:\\ q_t= k}} p(\mathbf{s}_t..\mathbf{s}_T|\mathbf{q},\Omega)}_{\tilde{\beta}_{k,t+1}b_k(\mathbf{s}_t)} + \underbrace{\sum_{\substack{\mathbf{q}\in Q:\\ q_t> k}} p(\mathbf{s}_t..\mathbf{s}_T|\mathbf{q},\Omega)}_{\tilde{\beta}_{k+1,t}}. && (5.31)
\end{aligned}$$

Die implizite Berechnung der Verantwortlichkeiten

Zur impliziten Berechnung der Verantwortlichkeiten können nun die eben definierten $\tilde{\alpha}_{kt}$ und $\tilde{\beta}_{kt}$ herangezogen werden. Zunächst gilt (s. Abschnitt 5.3):

$$r^i_{kt} = \frac{P(\mathbf{S}^i, q_t = k|\Omega)}{P(\mathbf{S}^i|\Omega)} = \frac{\sum\limits_{\substack{\mathbf{q}\in Q^i:\\ q_t=k}} \prod\limits_{t=1}^{T^i} b_{q_t}(\mathbf{s}^i_t)}{\sum\limits_{\mathbf{q}'\in Q^i} \prod\limits_{t=1}^{T^i} b_{q'_t}(\mathbf{s}^i_t)}. \qquad (5.32)$$

Benötigt werden also sowohl die Wahrscheinlichkeit $P(\mathbf{S}^i, q_t = k|\Omega)$, also die Wahrscheinlichkeit die Sequenz $\mathbf{S}^i$ zu beobachten und zum Zeitpunkt t im Zustand k zu sein, als auch die Produktionswahrscheinlichkeit $P(\mathbf{S}^i|\Omega)$.

Die implizite Berechnung der $P(\mathbf{S}^i|\Omega)$ stellt kein Problem dar, da die Produktionswahrscheinlichkeiten den $\tilde{\alpha}^i_{KT}$ und $\tilde{\beta}^i_{1,1}$ entsprechen. Der Zähler kann durch Ausnutzen der gleichzeitig berechneten Vorwärts- und Rückwärtsvariablen gemäß dem Schema in Abbildung 5.1 zusammengesetzt werden, sodass sich für die Zelle (k, t) der Vorwärtsanteil aus dem ersten Term von Gleichung 5.20 – nämlich $\tilde{\alpha}_{k,t-1}b_k(\mathbf{s}_t)$ – und der Rückwärtsanteil als $\tilde{\beta}_{k,t+1}$ ergibt.

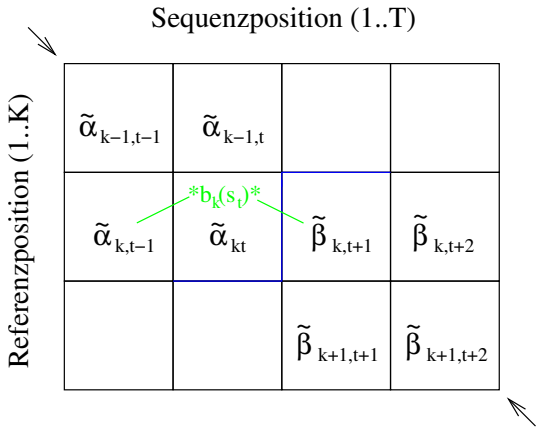

Abbildung 5.1.: Veranschaulichung des Prinzips zur Berechnung des Zählers der Verantwortlichkeiten.

Also erhalten wir die Verantwortlichkeiten implizit durch Berechnung von

$$r^i_{kt} = \frac{\tilde{\alpha}_{k,t-1}b_k(\mathbf{s}_t)\tilde{\beta}_{k,t+1}}{\tilde{\alpha}_{KT}}. \tag{5.33}$$

5.4. OMMbest

Bei diesem Modell beschränken wir die Berücksichtigung der Sequenzvektoren für die Parameterschätzung auf den besten (wahrscheinlichsten) Pfad der Sequenz bzgl. des Prototypen. Dies kann die Modellbildung vereinfachen.

Mit dem Grenzfall $\sigma \to 0$ in Gleichung 5.1 erhalten wir (nach der erforderlichen Logarithmierung) eine Dominanz des Distanzterms, welches dem Übergang zum Maximum und damit zum Viterbi-Algorithmus für die Pfadbestimmung entspricht.

Die Zuordnungen z_{kt} der Sequenzvektoren $\mathbf{s}_t$ auf die Prototypvektoren $\boldsymbol{\mu}_k$ sind in diesem Fall deterministisch, d.h. $z_{kt} \in \{0, 1\}$ und es muss gelten

$$\sum_{k=1}^{K} z_{kt} = 1.$$

Damit ergibt sich ein deterministischer Pfad durch das Modell.

Die OMMbest-Zielfunktion

Ziel der OMMbest-Modellbestimmung ist wiederum die Maximierung der Likelihood bzw. der Log-Likelihood, dass die Daten durch dieses Modell zustandegekommen sind.

Mit dem Pfad $\mathbf{q}$ ergibt sich:

$$P^*(\mathbf{S}^i|\Omega) = p(\mathbf{S}^i, \mathbf{q}^*|\Omega) = \max_{\mathbf{q}\in Q^i} p(\mathbf{S}^i, \mathbf{q}|\Omega) \tag{5.34}$$

$$\max \ln \prod_{i=1}^{N} P^*(\mathbf{S}^i|\Omega) = \max \sum_{i=1}^{N} \ln P^*(\mathbf{S}^i|\Omega). \tag{5.35}$$

Da σ fixiert ist, bleibt als einziger zu optimierender Parameter des OMMs Ω der Erwartungswert $\boldsymbol{\mu}_k$ der Emissionsdichten $b_k(\mathbf{s}_t)$.

Die optimale Zustandsfolge $\mathbf{q}^*$ kann mit Hilfe des Viterbi-Algorithmus effizient berechnet werden. Mit der Viterbivariable

$$v_t(k) = \max\{P(\mathbf{s}_1..\mathbf{s}_t, q_1..q_t|\Omega)|\mathbf{q} \in Q^i \text{ mit } q_t = k\} \tag{5.36}$$

lässt sich die Produktionswahrscheinlichkeit des besten Pfades durch ein dynamisches Programm lösen, wobei

$$v_1(k) = b_k(\mathbf{s}_1) \tag{5.37}$$

$$v_t(k) = \max_{k'} v_{t-1}(k')b_k(\mathbf{s}_t) \tag{5.38}$$

$$P^*(\mathbf{S}^i|\Omega) = \max_{k} v_T(k). \tag{5.39}$$

Dabei bezeichnet k' einen Vorgängerzustand. Durch gleichzeitiges Anlegen einer Rückverzeigerungsmatrix $\mathbf{\Phi}$ mit den jeweiligen Indizes der Maxima lässt sich nun die optimale Zustandsfolge $\mathbf{q}^*$ bestimmen.

Die Produktionswahrscheinlichkeitsdichte der optimalen Zustandsfolge $p^*(\mathbf{S}|\Omega)$ hängt also nur von der Länge der Sequenz T und den Emissionswahrscheinlichkeit $b_k(\mathbf{s}_t)$ ab.

Mit der Festlegung $P(\mathbf{q}|\Omega) = \frac{1}{M},\ \forall q \in Q$ gilt:

$$p^*(\mathbf{S}|\Omega) = P(\mathbf{q}^*|\Omega) \cdot p(\mathbf{S}|\mathbf{q}^*, \Omega) \tag{5.40}$$

$$= \frac{1}{M} \cdot \prod_{t=1}^{T} b_{q_t^*}(\mathbf{s}_t). \tag{5.41}$$

Setzen wir dies in die Zielfunktion ein und berücksichtigen, dass der pro Sequenz konstante Term $\frac{1}{M^i}$ bzgl. der Maximierung irrelevant ist, ergibt sich:

$$\max \sum_{i=1}^{N} \ln P^*(\mathbf{S}^i|\Omega) = \max \sum_{i=1}^{N} \sum_{t=1}^{T^i} \ln \left(\frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} \cdot e^{-\frac{1}{2\sigma^2} \cdot \|\mathbf{s}_t^i - \boldsymbol{\mu}_{q_t^*}\|^2} \right) \tag{5.42}$$

$$= \max -\sum_{i=1}^{N}\sum_{t=1}^{T^i} \|\mathbf{s}_t^i - \boldsymbol{\mu}_{q_t^*}\|^2 \text{ für } \sigma \to 0 \tag{5.43}$$

$$= \max -\sum_{i=1}^{N}\sum_{t=1}^{T^i}\sum_{k=1}^{K} z_{kt}^i \|\mathbf{s}_t^i - \boldsymbol{\mu}_k\|^2 \tag{5.44}$$

$$= \min \sum_{i=1}^{N}\sum_{t=1}^{T^i}\sum_{k=1}^{K} z_{kt}^i \|\mathbf{s}_t^i - \boldsymbol{\mu}_k\|^2, \tag{5.45}$$

wobei z_{kt}^i die Zuordnung von $\mathbf{s}_t^i$ auf $\boldsymbol{\mu}_k$ ist.

Die Optimierung der Zielfunktion lässt sich als iteratives Projektions-Regressions-Schema darstellen, wobei abwechselnd im Alignmentschritt die optimalen Zuordnungen z_{kt}^* und im Regressionsschritt die neuen Mittelwertvektoren $\boldsymbol{\mu}_k$ bestimmt werden (s.u.).

Betrachtung für $\sigma \to 0$

Die Grenzwertbetrachtung für $\sigma \to 0$ liefert nach Umformung des Klammerterms in Gleichung 5.42 zu

$$-d \cdot \ln(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2} \cdot \|\mathbf{s}_t^i - \boldsymbol{\mu}_{q_t^*}\|^2 \tag{5.46}$$

folgende Zusammensetzung der Konvergenzeigenschaften:

$$\lim_{\sigma \to 0} d \cdot \ln(\sqrt{2\pi}\sigma) = \infty \tag{5.47}$$

und

$$\lim_{\sigma \to 0} \frac{1}{2\sigma^2} \cdot \|\mathbf{s}_t^i - \boldsymbol{\mu}_{q_t^*}\|^2 = \infty \tag{5.48}$$

wobei Gleichung 5.47 nur reziprok exponentiell, Gleichung 5.48 jedoch mit quadratischer Geschwindigkeit gegen ∞ strebt. Daher dominiert der Distanzterm die Gleichung und der erste Term in Gleichung 5.46 (sowie $\frac{1}{2\sigma^2}$ als Konstante für fixiertes σ) kann für $\sigma \to 0$ vernachlässigt werden.

Regularisierte Zielfunktion

Kontinuierliche Daten sind oft verrauscht, d.h. die Werte streuen um einen Mittelwert. Außerdem kommt es bei parametrischen Methoden mit wenig Datenbeispielen schnell zu einer Überparametrisierung. Damit für beide Fälle eine Verschlechterung der Erkennungsleistung durch *overfitting* vermieden wird, benutzen wir eine Regularisierungsmethode. Mittels *Regularisierung* versuchen wir eine „weichere“ bzw. generellere Lösung des Minimierungsproblems zu finden. Dazu wird der Fehlerfunktion ein *Regularisierungsterm* $\lambda\phi(\mathbf{w})$ hinzugefügt, welcher die Lösung beschränkt. Dabei ist $\phi(\cdot)$ eine nichtnegative *Straffunktion* (*penalty*) (vgl. [CM98]).

Als Regularisierungsterm fungiert bei OMMbest der durch λ gewichtete quadratische Abstand der Mittelwertvektoren $\mathbf{w} = [(\boldsymbol{\mu}_1)^T, .., (\boldsymbol{\mu}_K)^T]^T \in \mathbb{R}^D = \mathbb{R}^{d \cdot K}$ zum mit α ska-

lierten Einheitsvektor, so dass sich

$$E = \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \sum_{k=1}^{K} z_{kt}^{i} \|\mathbf{s}_t^i - \boldsymbol{\mu}_k\|^2 + \lambda \|\alpha \cdot \mathbf{1}_D - \mathbf{w}\|^2 \right) \tag{5.49}$$

$$= \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{k=1}^{K} z_{kt}^{i} \|\mathbf{s}_t^i - \boldsymbol{\mu}_k\|^2 + \lambda \|\alpha \cdot \mathbf{1}_D - \mathbf{w}\|^2 \tag{5.50}$$

als regularisierte Zielfunktion ergibt. Der Vektor $\alpha \cdot \mathbf{1}_D$ fungiert dabei als Sollmittelpunkt. Durch die gleiche Bestrafung aller Prototypelemente ist der Strafterm über die Referenzpositionen hinweg eine konstante Funktion. E lässt sich ebenfalls als iteratives Schema aus Alignment- und Mittelwertberechnungsschritt ausdrücken.

Initialisierung

Die Initialisierung der Mittelwertvektoren vor dem ersten Alignmentschritt erfordert kein bestehendes multiples Alignment (wie es etwa bei PHMMs der Fall ist), sondern erfolgt allein anhand der gegebenen Sequenzen.

Dazu berechnen wir je Klasse $y_i \in Y$ aus jeder Sequenz $\mathbf{S}^i$ einen geglätteten, auf die erforderliche Länge K interpolierten Sequenzprototypen $\hat{\mathbf{S}}^i$. Der klassenspezifische Mittelwertvektor ist dann der Mittelwert aller zu einer Klasse gehörenden $\hat{\mathbf{S}}^i$.

Die Glättung erfolgt durch die Faltung der Sequenz mit einem Dreiecks-Glättungskern (*triangle smoothing kernel*) und anschließender Re-Normierung. Die Breite des Glättungskerns ist dabei anwendungsabhängig zu wählen. Sinnvolle Werte sind z.B. 5 bis 9 Einträge, für kleinere Werte findet keine ausreichende Glättung mehr statt, größere Werte benötigen erheblich mehr Rechenaufwand und zerstören unter Umständen wichtige Informationen. Die Interpolation führen wir mittels kubischer *splines* (stückweise Polynome dritten Grades) durch.

Alignment

Im Alignmentschritt bestimmen wir die optimalen Zuordnungen z_{kt}^* eines Sequenzvektors $\mathbf{s}_t$ auf einen Referenzvektor $\boldsymbol{\mu}_k$ durch die Minimierung von Gleichung 5.45 unter Variation der z_{kt}. Damit nicht alle Pfade explizit durchgerechnet werden müssen, bedienen wir uns des in den Gleichungen 5.37 bis 5.39 geschilderten dynamischen Programms. Dabei lässt sich das Maximum der Produktionswahrscheinlichkeit durch den negativen Logarithmus auf die Minimierung der Distanz übertragen. Das Alignmentschema ist anhand eines Beispiels in Abbildung 4.3 veranschaulicht.

Mittelwertberechnung

Für Gleichung 5.45 berechnen sich die Mittelwerte aus den zu den im Alignmentschritt gelernten optimalen Zuordnungen gehörigen Sequenzvektoren. Dazu muss die Fehlerfunktion in Gleichung 5.45 minimiert werden. Da sie quadratisch ist, gibt die Nullstelle

der Ableitung von E nach $\boldsymbol{\mu}_k$ das Optimum an:

$$\begin{aligned} \nabla_{\boldsymbol{\mu}_k} E &= \sum_{i=1}^{N} \sum_{t=1}^{T_i} z_{kt}^i \left(\boldsymbol{\mu}_k - \mathbf{s}_t^i \right) \stackrel{!}{=} \mathbf{0} \\ \Leftrightarrow \boldsymbol{\mu}_k &= \frac{\sum_{i=1}^{N} \sum_{t-1}^{T_i} z_{kt}^i \mathbf{s}_t^i}{\sum_{i=1}^{N} \sum_{t-1}^{T_i} z_{kt}^i}. \end{aligned} \tag{5.51}$$

Im regularisierten Fall (s. Gl. 5.50) schätzen wir zuerst α aus den wie oben berechneten $\boldsymbol{\mu}_k$ gemäß

$$\alpha = \frac{1}{d \cdot K} \sum_{k=1}^{K} \sum_{l=1}^{d} \mu_k^l. \tag{5.52}$$

Die neuen Mittelwerte ergeben sich aus dem Minimum von Gleichung 5.50. Dazu leiten wir wiederum die Zielfunktion jeweils nach $\boldsymbol{\mu}_k$ ab und bestimmen die Nullstelle:

$$\begin{aligned} \nabla_{\boldsymbol{\mu}_k} E &= \sum_{i=1}^{N} \sum_{t=1}^{T_i} z_{kt}^i \left(\boldsymbol{\mu}_k - \mathbf{s}_t^i \right) + \lambda(\boldsymbol{\mu}_k - \alpha \cdot \mathbf{1}_D) \stackrel{!}{=} \mathbf{0} \\ \Leftrightarrow \hat{\boldsymbol{\mu}}_k &= \frac{\sum_{i=1}^{N} \sum_{t=1}^{T_i} z_{kt}^i \mathbf{s}_t^i + \lambda\alpha \cdot \mathbf{1}_d}{\sum_{i=1}^{N} \sum_{t=1}^{T_i} z_{kt}^i + \lambda}. \end{aligned} \tag{5.53}$$

Dies lässt sich durch eine schrittweise Optimierung realiseren, wobei unmittelbar nach dem Alignmentschritt der Skalierungsparameter α und im Regressionsschritt dann die Prototypvektoren $\boldsymbol{\mu}_k$ bestimmt werden.

5.5. Klassifikation mit OMMs

Die Klassifikation neuer Sequenzen erfolgt für OMMs nach der maximalen Produktionswahrscheinlichkeit eines Modells Ω bzgl. der Sequenz. Dazu trainieren wir für jede Klasse $y_i \in Y$ ein Ω_{y_i}.

Für OMMall ergibt sich nach dem Alignieren der Testsequenz $\mathbf{S}$ gegen alle Modelle $\Omega_1, .., \Omega_M$

$$f(\mathbf{S}) = \arg\max_{y_i} P(\mathbf{S}|\Omega_{y_i}) = \arg\max_{y_i} (\alpha_{KT}^{y_i}). \tag{5.54}$$

Bei OMMbest entspricht $P(\mathbf{S}|\Omega_{y_i})$ der minimalen Distanzsumme des dynamischen Programms im Alignmentschritt:

$$f(\mathbf{S}) = \arg\min_{y_i} D(\mathbf{S}|\Omega_{y_i}). \tag{5.55}$$

D enthält dabei die kumulativen Distanzen.

5.6. Kerndichteschätzung und -klassifikation mit OMMs

Die OMMs bieten durch einen Spezialfall die Möglichkeit, Kerndichteschätzung und darauf aufbauende Methoden (s. Abschnitt 2.6) auf Sequenzen durchzuführen.

5.6.1. SSOMM

Ein Single-Sequence-Ordered-Means-Modell (SSOMM) einer Sequenz $\mathbf{S}^j$ bezüglich einer Sequenz $\mathbf{S}^i$ ist deren (optimales) paarweises Alignment und liefert die Produktionswahrscheinlichkeit

$$P(\mathbf{S}^j|\mathbf{S}^i) = \sum_{\mathbf{q}\in Q} P(\mathbf{S}^j|\mathbf{q},\mathbf{S}^i) \tag{5.56}$$

$$= \frac{1}{M}\sum_{\mathbf{q}\in Q}\prod_{t=1}^{T} b_{q_t}(\mathbf{s}_t^j). \tag{5.57}$$

Die Mittelwerte der Emissionsdichten entstammen hierbei der Sequenz $\mathbf{S}^i$. Somit beträgt die Prototypgröße für ein SSOMM $K = T^i$.

5.6.2. SSOMM-Kerndichteschätzung und -klassifikation

SSOMMs ermöglichen es, einen Sequenz-Dichtekern zu formulieren. Dabei fungieren die Produktionswahrscheinlichkeitsdichten als Kern $K(\mathbf{S}^j,\mathbf{S}^i)$. Gemäß Gleichung 2.14 berechnet sich die Kerndichte einer Sequenz $\mathbf{S}$ aus einer Menge von N Sequenzen dann folgendermaßen:

$$p(\mathbf{S}) = \sum_{i=1}^{N} \omega_i P(\mathbf{S}|\mathbf{S}^i) \tag{5.58}$$

$$\text{mit } \omega_i = \frac{1}{N}. \tag{5.59}$$

Dies kann man als Mischungsmodell mit N Komponenten auffassen.

Die Klassifikation mit SSOMMs kann dann analog zu Abschnitt 2.6.3 erfolgen.

5.6.3. Komplexitätsaspekte

Für die Dichteschätzung einer Menge von Sequenzen und die Klassifikation ist die Berechnung aller paarweisen SSOMM-Dichtekerne erforderlich. Dieser (in Abhängigkeit von der Datenmenge) quadratisch wachsende Aufwand ist für viele reale Anwendungen nicht praktikabel, stellt jedoch für kleine Probleme eine echte Alternative dar, da es sich um ein *nichtparametrisches Verfahren* handelt, d. h. es werden keine Modellparameter geschätzt. Der *OMMKDE* bleibt dabei trotzdem generativ.

Dennoch muss für die erfolgreiche Anwendung der Hyperparameter σ – welcher hier als Kernbreite fungiert – evaluiert werden. Mit den schon besprochenen Varianten der

OMMs ergeben sich auch für den *OMMKDE* verschiedene Szenarien. Diese sollen hier jedoch nicht ausführlich besprochen werden.

6. Feature-Alignment-Maschinen

6.1. Idee und Motivation

Feature-Alignment-Maschinen (FAMs) beschreiben eine Technik, die Algorithmen des maschinellen Lernens für Sequenzen nutzbar macht. Diese Technik projiziert durch eine adaptive Merkmalsauswahl die möglicherweise verschieden langen Sequenzen in einen expliziten Merkmalsraum, um sie dort weiterzuverarbeiten. Die Idee hinter der Merkmalsauswahl ist, nicht alle Abschnitte einer Sequenz bezüglich ihrer Wichtigkeit einer Fragestellung des maschinellen Lernens gleich zu bewerten, sondern nach Möglichkeit ausschließlich charakteristische Muster der Beispielsequenzen zu lernen. Das Ziel ist also gleichzeitiges Lernen von Modellen und Selektion von geeigneten Merkmalen von Sequenzen und ihre Transformation in den *Feature Space*.

Im Folgenden werden wir als erstes den Merkmalsraum der FAMs beschreiben (Abschnitt 6.2), dann im einzelnen die von uns entwickelten Algorithmen. Zu Beginn den *FAMmean*-Algorithmus (Abschnitt 6.3), der einen FAM-Mittelwertvektor aus den Daten schätzt, die *FASVMs* (Abschnitte 6.4 und 6.5), zwei SVM-Ansätzen auf Sequenzen, sowie die *PCA* im FAM-Merkmalsraum (Abschnitt 6.6), einer Hauptkomponentenanalyse auf Sequenzen. Der Abschnitt 6.7 beschreibt nochmal detailliert das Projektions-Regressions-Schema der Feature-Alignment-Maschinen. Abschließend werden wir in Abschnitt 6.8 erläutern, wie FAMs zur Klassifikation verwendet werden können.

6.2. Der Merkmalsraum

Das Zuordnungsprinzip

FAMs arbeiten nicht auf einzelnen Sequenzvektoren, sondern setzen Abschnitte der Sequenz zu *Sequenzmustervektoren* zusammen. Ein Sequenzmustervektor $\bar{\mathbf{s}}_t = [\mathbf{s}_t^T, .., \mathbf{s}_{t+L}^T]^T$ ist dabei aus $L \in \mathbb{N}$ aufeinanderfolgenden Sequenzvektoren zusammengesetzt, sodass sich für $\bar{\mathbf{S}}$ eine Folge $T - L + 1$ überlappender Sequenzabschnitte (die *Muster*) ergibt. T ist dabei die ursprüngliche Länge der Sequenz.

Für die eindimensionale Beispielsequenz

$$\mathbf{S} = [1 \quad 0 \quad 1 \quad -1 \quad 0 \quad 1]$$

und $L = 2$ ergibt sich also

$$\bar{\mathbf{S}} = \begin{bmatrix} 1 & 0 & 1 & -1 & 0 \\ 0 & 1 & -1 & 0 & 1 \end{bmatrix}$$

als Folge überlappender Sequenzabschnitte.

Die Zuordnung erfolgt dann nach dem Prinzip: „Genau ein Sequenzmustervektor wird auf jeden Referenzmustervektor zugeordnet, wobei keine Mehrfachzuordnungen auftreten dürfen“.

Die Prototypen

Ein Referenzvektor (o.a. Prototyp) $\mathbf{w} = [\bar{\mathbf{w}}_1^T, .., \bar{\mathbf{w}}_K^T]^T$ ist die Anordnung der *Referenzmustervektoren* $\bar{\mathbf{w}}_k$ für die Folgenlänge L. Dabei ist jedes $\bar{\mathbf{w}}_k$ aus L Teilmustervektoren $\mathbf{v}$ der Dimensionalität d zusammengesetzt. Die Prototypen können als Modell der zugrundeliegenden Daten interpretiert werden.

Der Merkmalsraum

Der Merkmalsvektor $\mathbf{x}$ einer Sequenz $\mathbf{S}$ setzt sich zusammen aus den – der gelernten Zuordnungen $\mathbf{Z}$ entsprechenden – Sequenzmustervektoren

$$\mathbf{x} = [\bar{\mathbf{s}}_{q_1}^T, .., \bar{\mathbf{s}}_{q_K}^T]^T. \tag{6.1}$$

Der Pfad $\mathbf{q}$ besitzt dabei K Komponenten, welche die zugeordneten Zeitpunkte $t = 1, .., T - L + 1$ der Sequenz darstellen. Die $\bar{\mathbf{s}}_{q_k}$ sind diejenigen Sequenzmustervektoren, für welche die Zuordnung $z_{kt} = 1$ ist. Die Anzahl der Prototypen K berechnet sich dabei aus $K = \frac{D}{L}$, wobei D eine zu spezifizierende Konstante ist.

Die Größe der Merkmalsvektoren einer Folgenlänge L berechnet sich damit aus der Dimensionalität der Sequenzen d, der Anzahl der überlappenden Sequenz- bzw. Referenzvektoren L und der Anzahl der Prototypen K mit $K \cdot L = D = const.$

Kombination verschiedener Folgenlängen

Mittels mehrerer $L^j \in L,\ j = 1, .., J$ (z.B. $L = \{2, 3, 4, 5\}$) kombinieren wir verschiedene Folgenlängen in einem Merkmalsvektor. So können auf einen Schlag Muster verschiedener Länge berücksichtigt werden. Die Merkmalsvektoren setzten sich dann aus $\mathbf{x} = [(\mathbf{x}^1)^T, .., (\mathbf{x}^J)^T]^T$ zusammen, die Prototypen werden in einem Vektor $\mathbf{w} = [(\mathbf{w}^1)^T, .., (\mathbf{w}^J)^T]^T$ zusammengefasst. Abbildung 6.1 veranschaulicht diesen Aufbau.

Insgesamt ergibt sich also die Dimension des Merkmalsraums $\mathcal{F}$ zu

$$d_{\mathcal{F}} = d \cdot \sum_{j=1}^{J} L^j \cdot K^j = d \cdot J \cdot D. \tag{6.2}$$

Im Folgenden betrachten wir zur besseren Übersichtlichkeit den Fall $J = 1$.

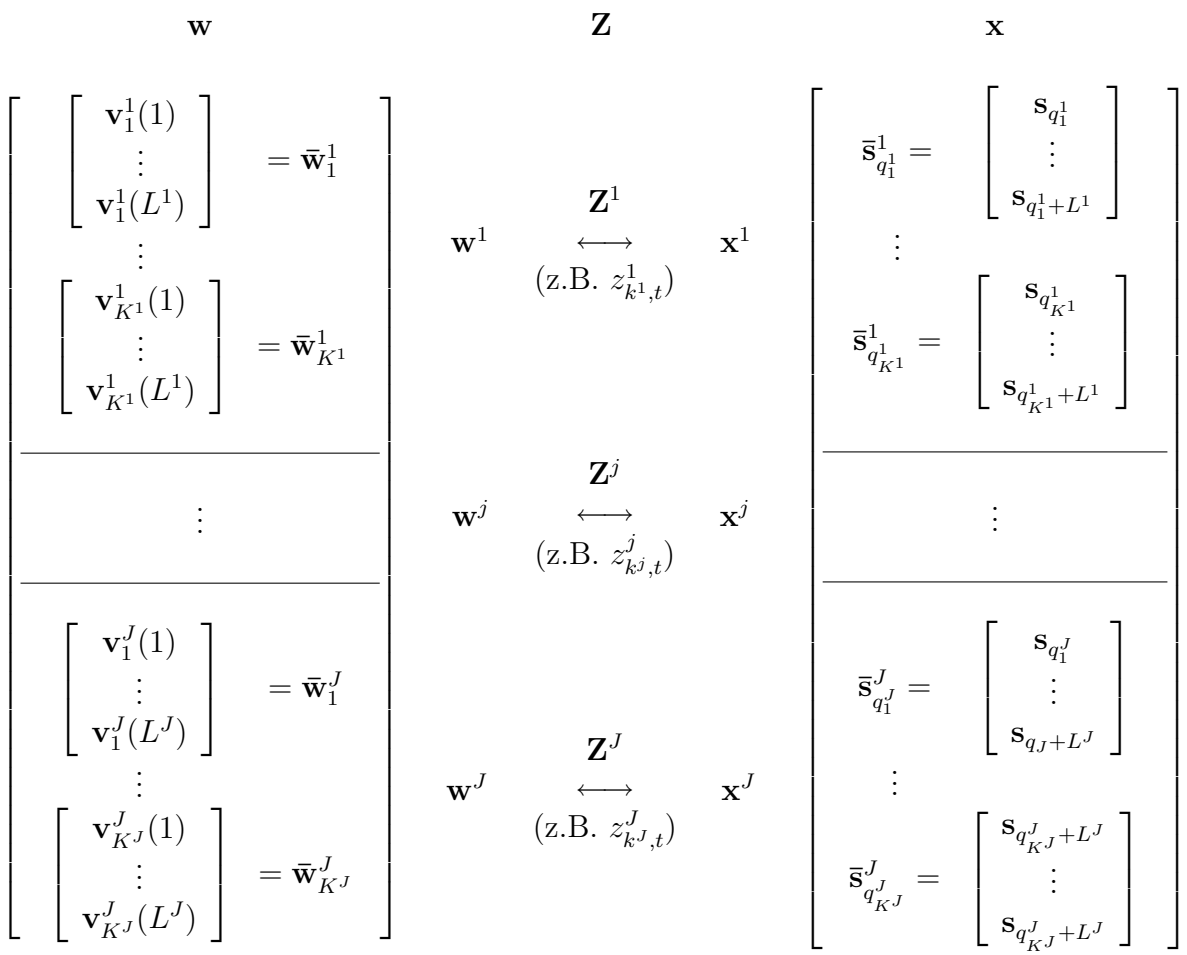

Abbildung 6.1.: Veranschaulichung des Zuordnungsprinzips der FAMs. Der Merkmalsraumprototyp $\mathbf{w}$ besteht aus einzelnen Referenzvektoren $\mathbf{w}^j$. Diese sind aus Referenzmustervektoren $\bar{\mathbf{w}}^j_k$ zusammengesetzt, die wiederum aus K^j Teilmustervektoren $\mathbf{v}$ der Dimension d bestehen. Den Referenzmustervektoren $\bar{\mathbf{w}}$ wird über die Zuordnungsmatrizen $\mathbf{Z}$ ein Sequenzmustervektor $\bar{\mathbf{s}}$ zugeordnet, welcher seinerseits aus Sequenzvektoren $\mathbf{s}$ der Dimension d besteht.

Heuristik für kurze Sequenzen

Das oben dargestellte Zuordnungsprinzip kann zu einer Situation führen, in der für einzelne Sequenzen mehr Referenzpositionen als Sequenzvektoren zur Verfügung stehen ($T < K$). Diese Sequenzen bedürfen einer speziellen Behandlung, da sonst Referenzvektorpositionen unbesetzt bleiben und die Sequenzen nicht mehr in den Merkmalsraum projizierbar sind.

Das naheliegende Vorgehen in einem solchen Fall ist, diese Sequenzen aus der Trainingsmenge zu entfernen und beispielsweise als falsch klassifizierte Testsequenzen einzustufen. Ein solches Vorgehen schränkt die Auswahl des Parameters K stark ein, was insbesondere bei Daten mit hoher Varianz der Sequenzlänge zu Problemen führen kann.

Schon bei sehr kleinem K würden viele Sequenzen den Testfehler im vorhinein erhöhen, was sich negativ auf die Gesamtperformanz auswirkt.

Besser ist es, die Sequenzen durch eine Heuristik künstlich auf die erforderliche Länge zu erweitern. Unser Ansatz hängt an den Anfängen und Enden der Sequenzen gleich viele *Sequenzmittelwertvektoren* $\mathbf{s}_m \in \mathbb{R}^d$ an (*Padding*). Diese Sequenzmittelwertvektoren berechnen sich gemäß

$$\mathbf{s}_m^i = \frac{1}{T} \sum_{t=1}^{T} \mathbf{s}_t^i. \tag{6.3}$$

Um eine Merkmalsauswahl überhaupt möglich zu machen, wählen wir die minimale Länge der Sequenzen als $T_{min} = K + \frac{K}{10}$. Die Sequenzen sind also 10% länger, als es die Anzahl der Prototyppositionen erfordert.

Natürlich sind noch andere Vorgehen und Heuristiken als die oben beschriebenen denkbar und im Einzelfall ist zu prüfen, ob die vorhandene Datenbasis nicht eine eigene Variante nahelegt.

6.3. Feature-Alignment Mean

Dem *FAMmean*-Algorithmus liegt einer der einfachsten Prototypenalgorithmen zugrunde (vgl. Abschnitt 2.4). Der Prototyp $\mathbf{w}$ ist in diesem Fall der Mittelwertvektor der in den Merkmalsraum projizierten Sequenzen.

Die Zielfunktion des FAMmean ist

$$E_{FAMmean} = \min \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{k=1}^{K} z_{kt}^i \|\bar{\mathbf{s}}_t^i - \bar{\mathbf{w}}_k\|^2 \tag{6.4}$$

$$= \min \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{k=1}^{K} z_{kt}^i \left(\|\bar{\mathbf{s}}_t^i\|^2 - 2\bar{\mathbf{s}}_t^i \cdot \bar{\mathbf{w}}_k + \|\bar{\mathbf{w}}_k\|^2 \right) \tag{6.5}$$

$$\text{u.d.B.d} \qquad \|\bar{\mathbf{s}}_t^i\| = \|\bar{\mathbf{w}}\| = 1/\sqrt{K}. \tag{6.6}$$

Die Nebenbedingungen motivieren sich zum einen durch eine erforderliche Vergleichbarkeit des Scores im Projektionsschritt (vgl. Abschnitt 6.7), zum anderen liegen die Merkmalsvektoren der Sequenzen auf einer Hypersphäre, haben also die einheitliche euklidische Länge 1. Um dies zu gewährleisten, müssen die einzelnen Sequenzmustervektoren $\bar{\mathbf{s}}_t^i$ auf Länge $1/\sqrt{K}$ normiert werden, denn schließlich werden die Merkmalsvektoren $\mathbf{x}$ aus K Sequenzmustervektoren $\bar{\mathbf{s}}$ zusammengesetzt.

Diese Zielfunktion lässt sich analytisch nicht lösen, so dass wir auf ein iteratives Projektions-Regressions-Schema zurückgreifen müssen.

Zunächst werden im Projektionsschitt die Zuordnungen z_{kt}^i optimiert. Aufgrund der Nebenbedingung haben die $\|\bar{\mathbf{s}}_t^i\|^2$ keinen Einfluss im Projektionsschritt und durch das Zuordnungsprinzip kommt jedes $\bar{\mathbf{w}}_k$ genau einmal vor, so dass die $\|\bar{\mathbf{w}}_k\|^2$ ebenfalls weg-

fallen. Für den Projektionsschritt ergibt sich also

$$\hat{\mathbf{Z}} = \max_{\mathbf{Z}} \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{k=1}^{K} z_{kt}^{i} (\bar{\mathbf{s}}_t^i \cdot \bar{\mathbf{w}}_k). \tag{6.7}$$

Im Regressionsschritt werden nun die neuen Zuordnungen verwendet, um die $\bar{\mathbf{w}}_k$ zu berechnen:

$$\begin{aligned} \nabla_{\bar{\mathbf{w}}_k} E_{FAMmean} &= \sum_{i=1}^{N} \sum_{t=1}^{T_i} \hat{z}_{kt}^{i} \left(\bar{\mathbf{w}}_k - \bar{\mathbf{s}}_t^i \right) \stackrel{!}{=} \mathbf{0} \\ \Leftrightarrow \hat{\bar{\mathbf{w}}}_k &= \frac{\sum_{i=1}^{N} \sum_{t=1}^{T_i} \hat{z}_{kt}^{i} \bar{\mathbf{s}}_t^i}{\sum_{i=1}^{N} \sum_{t=1}^{T_i} \hat{z}_{kt}^{i}} \qquad (6.8) \\ \Leftrightarrow \hat{\mathbf{w}} &= \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}^i. \qquad (6.9) \end{aligned}$$

Die letzte Umformung gilt aufgrund des Zuordnungsprinzips ($\sum_{i=1}^{N} \sum_{t=1}^{T_i} \hat{z}_{kt}^{i} = N$).

6.4. Feature-Alignment SVMs

Der *FASVM*-Algorithmus ermöglicht es, die etablierten SVMs, welche zur Klassifikation, Regression und Dichteschätzung (siehe Abschnitt 2.7) benutzt werden, auf Sequenzen zu verwenden. Der in diesem Abschnitt geschilderte Ansatz verwendet eine Ein-Klassen-SVM, um den Prototypen $\mathbf{w}$ zu schätzen. Die Ein-Klassen-SVM verfolgt einen unüberwachten Ansatz des maschinellen Lernens.

Die primale Zielfunktion der ν-SVM in der Ein-Klassen-Variante mit linearem Kern lautet:

$$O_{FASVM} = \min_{\mathbf{w}, \boldsymbol{\xi}, \rho} \quad \frac{1}{2} \|\mathbf{w}\|^2 + \frac{1}{\nu N} \sum_{i=1}^{N} \xi^i - \rho \tag{6.10}$$

$$\text{u.d.B.d} \quad (\mathbf{w} \cdot \mathbf{x}^i) \geq \rho - \xi^i, \quad \xi^i \geq 0 \tag{6.11}$$

wobei

$$\mathbf{w} \cdot \mathbf{x}^i = \sum_{t=1}^{T} \sum_{k=1}^{K} z_{kt}^{i} (\bar{\mathbf{s}}_t^i \cdot \bar{\mathbf{w}}_k). \tag{6.12}$$

Der FASVM-Prototyp

Der *FASVM*-Prototyp, also der Gewichtsvektor der SVM, kann in diesem speziellen Fall als ein stabiler Mittelwertvektor interpretiert werden.

In hochdimensionalen Räumen ist die Schätzung von Mittelwertvektoren problematisch (s. [Ste56]). Als Lösung ist die Verwendung des Ausreißern gegenüber als robust

bekannten Medians denkbar, besser jedoch ist es, nur einen Anteil der Beispieldaten, welche im Inneren des Datenclusters liegen, zur Berechnung von $\mathbf{w}$ zu benutzen.

Ganz Ähnliches leistet die ν-SVM. Auch hier wird nur ein Teil der Daten zur Schätzung der Prototypen herangezogen. Dieser Anteil kann durch den Hyperparameter ν eingestellt werden (s. Abschnitt 2.7.6).

Deutlich wird dies in einem zweidimensionalen Beispiel. In Abbildung 6.2 links wurden nur 10% aller Daten zur Berechnung von $\mathbf{w}$ herangezogen, in Abbildung 6.2 rechts 90%. Der Einfluss auf $\mathbf{w}$ wird unmittelbar deutlich.

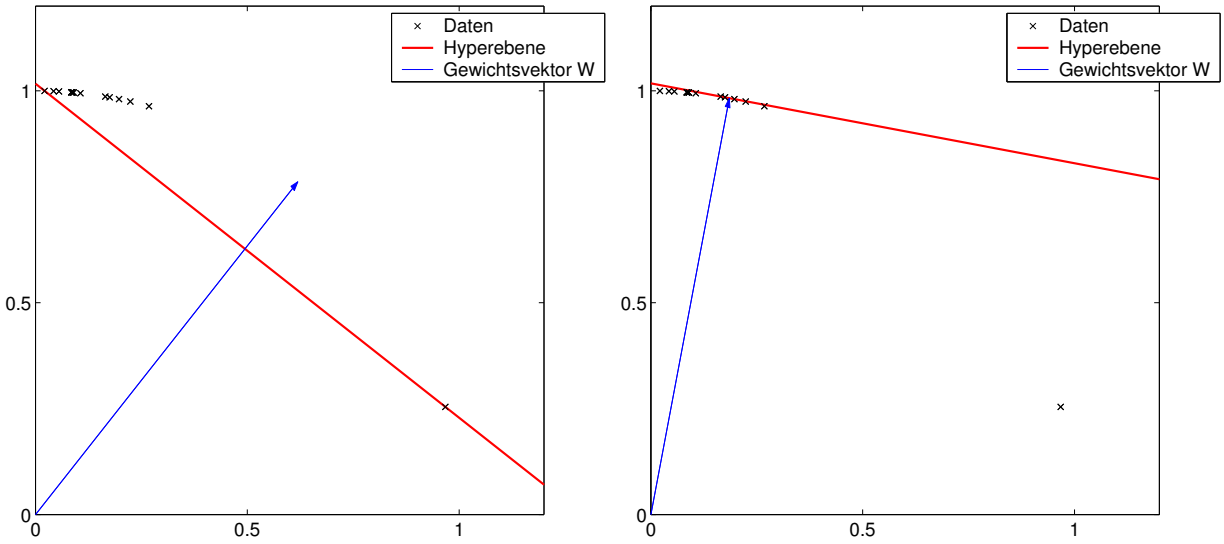

Abbildung 6.2.: Visualisierung einer Ein-Klassen-SVM auf zufälligen Beispieldaten, links ist $\nu = 0.1$, rechts ist $\nu = 0.9$. Der Gewichtsvektor ist auf euklidische Länge 1 normiert.

6.5. Diskriminative Feature-Alignment SVMs

Als Erweiterung des unüberwachten Lernens der Ein-Klassen-SVM bietet es sich an, eine überwachte Variante zu entwickeln. Dieses Vorgehen ermöglicht diskriminatives Lernen von Sequenzen zur Klassifikation. Zur Bestimmung des Prototypen werden nicht nur die Beispiele einer Klasse herangezogen, sondern es werden auch Sequenzen von anderen Klassen verwendet, um die *Unterschiede* zweier oder mehrerer Sequenzklassen zu lernen. Da der SVM-Ansatz historisch auf der Klassifikation beruht, ist es naheliegend, diese Fähigkeiten auch für FAMs zu nutzen.

Das Vorgehen entspricht dabei dem der ν-SVM, jedoch werden alle Beispielsequenzen gegen den Prototypen aligniert und im Regressionsschritt eine Hyperebene zwischen die positive und die negative Klasse der Merkmalsraumprojektionen der Sequenzen gelegt (vgl. Abb. 6.3).

6.6. PCA im FAM-Merkmalsraum

Um eine Hauptkomponentenanalyse von Sequenzen durchzuführen, berechnen wir zunächst eine Merkmalsraumprojektion mit Hilfe der FAMs. Auf den so entstandenen $\mathbf{x}^i$ kann dann eine Hauptkomponentenanalyse durchgeführt werden (vgl. Abschnitt 2.8).

6.7. Projektions-Regressions-Schema

Die optimalen Prototypen $\mathbf{w}$ sind nicht analytisch bestimmbar. Daher ist eine iterative Optimierung mit Projektions- und Regressionsschritt erforderlich.

Im Projektionsschritt werden die Sequenzen durch das optimale Alignment mit dem Prototypen in den Merkmalsraum abgebildet. Dort kann dann durch den Regressionsschritt die Schätzung des neuen Prototypen stattfinden.

Ziel der Optimierung ist die Minimierung einer Fehlerfunktion E (*FAMmean*) bzw. einer Zielfunktion O (*FASVM*).

6.7.1. Initialisierung

Die Prototypen $\mathbf{w}_{init}$ für den ersten Projektionsschritt werden aus den Trainingssequenzen bestimmt. Die Initialisierung erfolgt analog zu Abschnitt 5.4.

6.7.2. Die Projektion

Im Projektionsschritt werden sequenzweise die Zuordnungen $z_{kt} \in \{0, 1\}$ bestimmt und die Merkmalsvektoren $\mathbf{x}$ anhand dieser zusammengesetzt. Dies erfolgt j mal pro Sequenz, also für jede Sequenzmusterlänge L^j einmal, sodass sich die nach Abschnitt 6.2 erforderlichen Merkmalsvektoren ergeben.

Die Bestimmung des optimalen Pfades $\mathbf{q}$ erfolgt mittels Maximierung eines *Score* $S(\mathbf{w}, \mathbf{x})$, welcher die Ähnlichkeit einer Folge von Sequenzmustervektoren $\bar{\mathbf{S}} \in \mathbb{R}^{d \cdot L \times T-L+1}$ zum zu einer Matrix umgeformten Referenzvektor $\mathbf{W} \in \mathbb{R}^{d \cdot L \times K}$ aus dem vorhergehenden Regressionschritt angibt. Der kumulative Score $S(\mathbf{W}, \bar{\mathbf{S}})$ berechnet sich dabei gemäß folgendem Dynamic-Programming-Schema:

$$
\begin{aligned}
S(\mathbf{W}, \bar{\mathbf{S}}) &= score(K, T), \\
score(k, t) &= \max\{score(k, t-1) + \hat{P}_{\mathbf{W},\bar{\mathbf{S}}}(k, t),\ score(k-1, t)\}, \qquad (6.13) \\
score(0, 0) &= score(0, 1) = 0, \\
score(1, 0) &= -\infty, \\
\hat{P}_{\mathbf{W},\bar{\mathbf{S}}}(k, t) &= \bar{\mathbf{w}}_k \cdot \bar{\mathbf{s}}_t. \qquad (6.14)
\end{aligned}
$$

$\hat{P}_{\mathbf{W},\bar{\mathbf{S}}}$ ist die Ähnlichkeitsmatrix (*proximity matrix*) von $\mathbf{W}$ zu $\bar{\mathbf{S}}$. Da diese auf dem Skalarprodukt $\bar{\mathbf{w}}_k \cdot \bar{\mathbf{s}}_t$ beruht, müssen die Sequenzmustervektoren $\bar{\mathbf{s}}_t$ eine einheitliche euklidische Länge haben, damit das Skalarprodukt ein valides Ähnlichkeitsmaß darstellt.

Dies bewerkstelligen wir durch eine vorangestellte Normierung von allen $\bar{\mathbf{s}}_t^i$ auf Länge $1/\sqrt{K}$.

Die charakteristische Zuordungsfunktion z_{kt} wird dann über eine Rückverzeigerungsmatrix $\mathbf{\Phi}$ berechnet. $\mathbf{\Phi}$ bestimmen wir während des DP durch

$$\mathbf{\Phi}(k,t) = \arg\max\{score(k,t-1) + \hat{P}_{\mathbf{W},\bar{\mathbf{S}}}(k,t),\ score(k-1,t)\}$$

in Gleichung 6.13.

Dieses Alignmentpinzip minimiert die globalen Zielfunktionen der verschiedenen FAM-Algorithmen. So wird in Gleichung 6.7 deutlich, dass eine Maximierung des Skalarproduktes einer Minimierung der Fehlerfunktion des *FAMmean* entspricht.

Für die *FASVM* gilt Ähnliches. Die Nebenbedingung der globalen Zielfunktion sieht für jeden Datenpunkt vor, dass das Skalarprodukt aus Gewichtsvektor und Datenpunkt größer oder gleich der vom Abstand der Hyperebene zum Ursprung abgezogenen Hilfsvariable ξ^i des Datenpunkts ist. Letzteres wird, unter Berücksichtigung des Vorzeichens, in der Zielfunktion über eine Summenbildung maximiert. Damit wird auch das Skalarprodukt selber maximiert und die Integration des Alignmentschritts in die Zielfunktion erkennbar.

Das Alignment entspricht somit einer Transformation aus dem Raum $\mathbb{R}^{d\times T}$ in den $\mathbb{R}^{d\cdot J\cdot D}$. Man kann diese Transformation als Merkmalsselektion mit Randbedingungen (Ordnungserhaltung, Verzicht auf Mehrfachzuordnungen) interpretieren.

6.7.3. Die Regression

Nachdem alle Sequenzen $\mathbf{S}^i,\ i = 1,..,N$ gegen den Prototypen $\mathbf{w}$ aligniert wurde, bilden die so entstandenen $\mathbf{x}^i$ den Trainingsdatensatz aus Merkmalsvektoren derselben Dimensionalität. Hier können Standardmethoden des maschinellen Lernens zur Schätzung neuer Prototypen $\mathbf{w}$ herangezogen werden.

FAMmean

Der *FAMmean*-Algorithmus berechnet aus den Merkmalsvektoren durch

$$\mathbf{w} = \frac{1}{N}\sum_{i=1}^{N}\frac{\mathbf{x}^i}{||\mathbf{x}^i||} \tag{6.15}$$

einen Mittelwertvektor $\mathbf{w}$.

Die Optimierung des *FAMmean*-Modells endet, sobald die relative Veränderung der Spur der Kovarianzmatrix der Merkmalsvektoren mit $\mathbf{X} = [\mathbf{x}^1,..,\mathbf{x}^N]$:

$$spur(\frac{1}{N}\mathbf{X}\cdot\mathbf{X}^T - \mathbf{w}\mathbf{w}^T) \tag{6.16}$$

unter einem Schwellwert (z.B. 0,1%) liegt.

FASVM

Für die Berechung von $\mathbf{w}$ benutzen wir hier eine Ein-Klassen-SVM (ν-SVM, s. Abschnitt 2.7.6). Das Training ist beendet, wenn die Zielfunktion (*objective*) der SVM konvergiert, d.h. sich nicht mehr ändert.

Beschleunigung des Alignments

Der Projektionsschritt lässt sich für die *FASVM* beschleunigen, indem ausschließlich diejenigen Sequenzen neu aligniert werden, welche nach dem vorhergehenden Regressionsschritt Supportvektoren sind bzw. sich unterhalb der oberen Grenze des Korridors der SVM (*margin*) befinden. Das sind diejenigen Sequenzen $\mathbf{S}^i$, für die

$$\mathbf{w} \cdot \mathbf{x}^i + b \leq 1 \tag{6.17}$$

gilt. Je nach Spärlichkeit der Lösung der SVM kann dies die Reduktion auf ein Trainingsbeispiel bedeuten.

Diskriminative FASVM

Die Berechnung führen wir mit einer 2-Klassen-SVM durch. Im Multiklassenfall werden die Trainingsdaten aller Gegenklassen zu einer Menge (negative Klasse) zusammengefasst.

Alignment der zweiten Klasse

Im Projektionsschritt werden möglicherweise Vektoren der negativen Klasse über den Rand

$$\{\mathbf{x} | \mathbf{w} \cdot \mathbf{x} + b = -1\} \tag{6.18}$$

in Richtung der positiven Klasse bewegt. Damit die Zielfunktion der SVM nicht unzulässigerweise verändert wird, müssen diese Vektoren wieder bis zu diesem Rand zurückbewegt werden. Diese Rückbewegung ist in Abbildung 6.3 veranschaulicht. Dabei ist zu beachten, dass Datenpunkte, die mittels der Hilfsvariablen schon vorher die Randbedingung verletzten, nicht berücksichtigt werden. Man erkennt, dass außer der aus Gleichung 6.18 folgenden Bedingung

$$\mathbf{w} \cdot \mathbf{x} + b = -1 \tag{6.19}$$

auch

$$\mathbf{x} = \frac{(1-a) \cdot \mathbf{x}_o + a \cdot \mathbf{x}_n}{\|(1-a) \cdot \mathbf{x}_o + a \cdot \mathbf{x}_n\|} \tag{6.20}$$

gelten muss, damit $\mathbf{x}$ auf dem Hyperkugelabschnitt zwischen dem alten Merkmalsvektor $\mathbf{x}_o$ und dem zu weit bewegten $\mathbf{x}_n$ liegt. Der Parameter $a \in]0, 1[$ legt dabei die genaue Position fest und muss durch Einsetzten von Gleichung 6.20 in Gleichung 6.19 bestimmt werden:

$$\mathbf{w}\left((1-a)\mathbf{x}_o + a\mathbf{x}_n\right) \;=\; (-1-b) \cdot \|(1-a)\mathbf{x}_o + a\mathbf{x}_n\|. \tag{6.21}$$

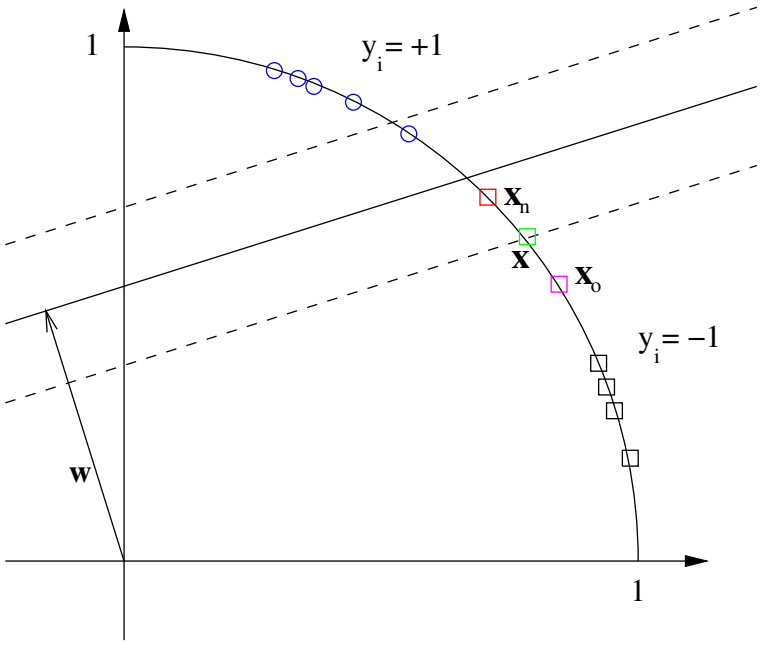

Abbildung 6.3.: Veranschaulichung der Rückbewegung der im Alignmentschritt zu weit bewegten Datenbeispiele der negativen Klasse (s. Text). Die durchgezogene Linie verkörpert die Hyperebene, die gestrichelten Linien stellen die Ränder $\{\mathbf{x}|\mathbf{w}\cdot\mathbf{x}+b=\pm 1\}$ dar.

Wir quadrieren beide Seiten, um die Norm auf der rechten Seite zu beseitigen. Die linke Seite rechnet sich dann folgendermaßen aus:

$$\begin{aligned} & \|\mathbf{w}\|^2\left[((1-a)\mathbf{x}_o)^2+2(1-a)\mathbf{x}_o a\mathbf{x}_n+(a\mathbf{x}_n)^2\right] \\ = \;& a^2\mathbf{w}(\mathbf{x}_o-\mathbf{x}_n)^2+2a\|\mathbf{w}\|^2\mathbf{x}_o(\mathbf{x}_n-\mathbf{x}_o)+\|\mathbf{w}\|^2\|\mathbf{x}_o\|^2. \end{aligned}$$

Bei der rechten Seite ergibt sich analog:

$$\begin{aligned} & \underbrace{(b^2+2b+1)}_{\bar{b}}\left[((1-a)\mathbf{x}_o)^2+2(1-a)\mathbf{x}_o a\mathbf{x}_n+(a\mathbf{x}_n)^2\right] \\ = \;& a^2\bar{b}(\mathbf{x}_o-\mathbf{x}_n)^2+2a\bar{b}\mathbf{x}_o(\mathbf{x}_n-\mathbf{x}_o)+\bar{b}\|\mathbf{x}_o\|^2. \end{aligned}$$

Durch Subtrahieren und Umformen nach a erhalten wir die quadratische Gleichung

$$\begin{aligned} a^2(\|\mathbf{w}\|^2-\bar{b})(\mathbf{x}_o-\mathbf{x}_n)^2+2a(\|\mathbf{w}\|^2-\bar{b})\mathbf{x}_o(\mathbf{x}_n-\mathbf{x}_o)+(\|\mathbf{w}\|^2-\bar{b})\|\mathbf{x}_o\|^2 &= 0 \\ \Leftrightarrow a^2+2a\frac{\mathbf{x}_o(\mathbf{x}_n-\mathbf{x}_o)}{(\mathbf{x}_o-\mathbf{x}_n)^2}+\frac{\|\mathbf{x}_o\|^2}{(\mathbf{x}_o-\mathbf{x}_n)^2} &= 0 \\ \Leftrightarrow a^2-2a\underbrace{\frac{\mathbf{x}_o\mathbf{x}_d}{\|\mathbf{x}_d\|^2}}_{p}+\underbrace{\frac{\|\mathbf{x}_o\|^2}{\|\mathbf{x}_d\|^2}}_{q} &= 0 \end{aligned}$$

wobei $\mathbf{x}_d = \mathbf{x}_o - \mathbf{x}_n$ ist. Diese hat die Lösungen

$$a_{1,2} = p \pm \sqrt{(2p)^2 - q}. \tag{6.22}$$

Nur eine davon liegt im erforderlichen Intervall $]0, 1[$. $\mathbf{x}$ ergibt sich gemäß Gleichung 6.20.

6.8. Klassifikation mit FAMs

Klassifikation mit FAMs realisieren wir, indem für jede Klasse $y_i \in Y$ ein eigener Prototyp $\mathbf{w}_{y_i}$ aus den gelabelten Beispielsequenzen berechnet wird. Die Entscheidungsfunktion $f(\mathbf{S}_{test})$ ordnet die ungelabelte Sequenz $\mathbf{S}_{test}$ der Klasse y_i zu, deren $\mathbf{w}_{y_i}$ die größte Ähnlichkeit bzw. den geringsten Abstand zur Merkmalsraumprojektion $\mathbf{x}_{test}$ der Sequenz hat. Im Detail bedeutet das für die Algorithmen:

FAMmean

Wie in Abschnitt 2.4 motiviert, klassifizieren Mittelwertklassifikatoren nach geringstem Abstand. Die Entscheidungsfunktion ist

$$\begin{aligned} f(\mathbf{S}_{test}) &= \arg\min_{y_i} ||\mathbf{x}_{test} - \mathbf{w}_{y_i}|| && (6.23) \\ &= \arg\max_{y_i} (\mathbf{x}_{test} \cdot \mathbf{w}_{y_i}). && (6.24) \end{aligned}$$

Dies lässt sich analog zu Gleichung 6.4ff umstellen.

FASVM

Die Entscheidungsfunktionen der *FASVMs* orientiert sich direkt am maximalen Skalarprodukt:

$$f(\mathbf{S}_{test}) = \arg\max_{y_i} \frac{\mathbf{w}_{y_i} \cdot \mathbf{x}_{test}}{\|\mathbf{w}_{y_i}\|}. \tag{6.25}$$

Um eine Gleichbewertung der verschieden langen $\mathbf{w}_{y_i}$ zu gewährleisten, müssen diese allerdings noch auf gleiche Länge normiert werden.

7. Evaluation

7.1. Datensätze

7.1.1. EEG-Daten

Das Interesse an der Funktionsweise des menschlichen Gehirns besteht schon sehr lange. Eine Möglichkeit, die Aktivität des Gehirns zu messen, ohne den Menschen dabei zu verletzen (*nichtinvasiv*), besteht darin, ein *Elektroenzephalogramm* (EEG, *Enzephalon* (griech.): das Gehirn) aufzunehmen.

Beim EEG werden Spannungsschwankungen auf der Kopfhaut mittels einer oder mehrerer Elektroden abgeleitet und – entweder mittels eines Plotters auf Papier oder digital – aufgezeichnet. Die elektrischen Hirnströme entstehen dabei durch die Weiterleitung von im (Neo-)Kortex entstehenden Aktionspotentialen großer Nervenzellverbände (einige Tausend bis mehrere Mio. Neuronen). Der Neokortex ist der stammesgeschichtlich jüngste Teil des Gehirns und der am stärksten differenzierte Teil der Großhirnrinde. Alle höheren kognitiven Leistungen sind dort lokalisiert (s. [Mül03]). Die Amplitude der abgeleiteten Potentiale (ohne Muskelartefakte) liegt im Bereich von ca. -100 bis 100 μV.

Für das EEG sind, neben der Amplitude, die Freqenzen, mit denen die Potentiale auftreten, wichtig. Man unterscheidet beim Menschen verschiedene Frequenzbänder (s. Tab. 7.1), denen unterschiedliche Bedeutungen beigemessen werden. Da die Ableitung nicht direkt, sondern an der Hautoberfläche erfolgt, werden beim EEG auch störende Muskelaktivitäten (*Artefakte*) mitaufgezeichnet.

Frequenzband	Frequenzbereich	Bedeutung
Delta	0.5 - 4 Hz	Tiefschlaf, Übergang vom Wach- in den Schlafzustand
Theta	4 - 7 Hz	Traumschlaf, Gedächtnis
Alpha	8 - 13 Hz	Grundrhythmus des ruhenden Gehirns
Beta	13 - 30 Hz	Sinnesreize und geistige Tätigkeit
Gamma	$>$ 30 Hz	Zusammenhang mit kognitiven Prozessen vermutet

Tabelle 7.1.: Einteilung der Frequenzbereiche des EEGs, nach [Zsch95].

Interessante Bestandteile des EEG-Signals stellen die *ereigniskorrelierten Potentiale* (*event related potentials*, ERP) dar. ERPs sind Reaktionen auf (optische, akustische etc.) Reize bzw. Ergebnisse interner Reizverarbeitung. Sie werden – aufgrund ihrer meist verhältnismäßig schwachen Amplitude – durch Mittelung über mehrere gleichartige Aufzeichnungen isoliert. Diese Analysetechnik vernichtet jedoch viel potentiell wertvolle Information und kann nicht in Echtzeit angewandt werden.

Das menschliche Gehirn lässt sich bzgl. kognitiver Fähigkeiten in wichtige Areale aufteilen. So findet z.B. die Sprachwahrnehmung und -produktion im vorderen Teil des Gehirns (*frontal* und *temporal*) – bei den meisten Menschen auf der linken Seite –, die visuelle Perzeption dagegen im hinteren Teil des Gehirns (*okzipital*) statt.

Probleme bei der Aufzeichnung des EEG sind oben genannte Muskelaktivitäten, die sich (auch aufgrund der Entfernung zu den ursächlichen Potentialen) durch ein schlechtes Signal-Rausch-Verhältnis und Artefakte bemerkbar machen sowie die Delokalisierung, da pro mm^2 Kopfhaut ca. 1 Mio. Neuronen Potentiale verursachen.

Eine zuverlässige Analyse ereigniskorrelierter Potentiale ist für viele praktische Einsatzbereiche (z.B. klinische Neurologie) erforderlich. Unsere Ansätze sollen die Ersetzbarkeit bisheriger Techniken (wie z.B. der Mittelung) durch Methoden des maschinellen Lernens untersuchen und die Analysen schneller und genauer machen.

Ursprung

Die EEG-Daten stammen aus Experimenten zur Sprachverarbeitung von Horst M. Müller und Sabine Weiss, welche im Rahmen des SFB 360 („Kortikale Repräsentation von Sprache “) stattfanden. Versuchspersonen waren 25 weibliche deutschsprachige Personen. Das Ziel des Versuchs war das Merken von Wörtern unterschiedlicher Kategorien: Abstrakta, Konkreta, Eigennamen und Verben. Sprachwahrnehmung ist ein äußerst komplexer und bislang nicht aufgeklärter kognitiver Prozess, der nicht auf ein spezielles Sprachorgan im Gehirn zurückzuführen ist (vgl. [Mül03]).

Die Präsentation der Wörter einer Bedingung erfolgte randomisiert (d.h. in zufälliger Reihenfolge), aber in Blöcken gleicher Bedingungen. Dies verringert die Gefahr, die – aufgrund der sich bei EEG-Aufzeichnungen über die Zeit verändernden – elektrischen Eigenschaften als diskriminative Merkmale zu interpretieren und gleiche auftretende Assoziationen von vorher gehörten Wörtern mitzulernen.

Die Ableitung erfolgte durch 19 Elektroden nach dem international standardisierten 10-20-System, außerdem enthält das Signal Augenbewegungs- (EOG) und Triggerinformationen (s. Abb. 7.1). Die Abtastrate beträgt 256 Hz, wobei ein Bandpass-Filter (0.3 - 35 Hz) und ein *Notch*-Filter um 50 Hz (zur Reduktion der Wechselspannungseinstreuungen) zum Einsatz kamen. Die Daten sind in 16-bit-Binärrepräsentation abgespeichert. Der qualitative Verlauf der aufgezeichneten EEG-Signale ist in Abbildung 7.1 nachzuvollziehen.

Mehrere Versuchspersonen (Vp) haben wir vor der Merkmalsextraktion nach Absprache mit H. M. Müller und S. Weiss aufgrund korrupter Daten (Vp 8), Abbruch eines oder mehrerer Versuche (Vp 4, 5, 10, 23), starker Muskelaktivität bzw. starker Augenartefakte (Vp 19, 21, 22) komplett entfernt.

Transformation

Nach Isolierung der Teilversuche konvertierten wir die Daten zur einfacheren Weiterverarbeitung in das Matlab-Format.

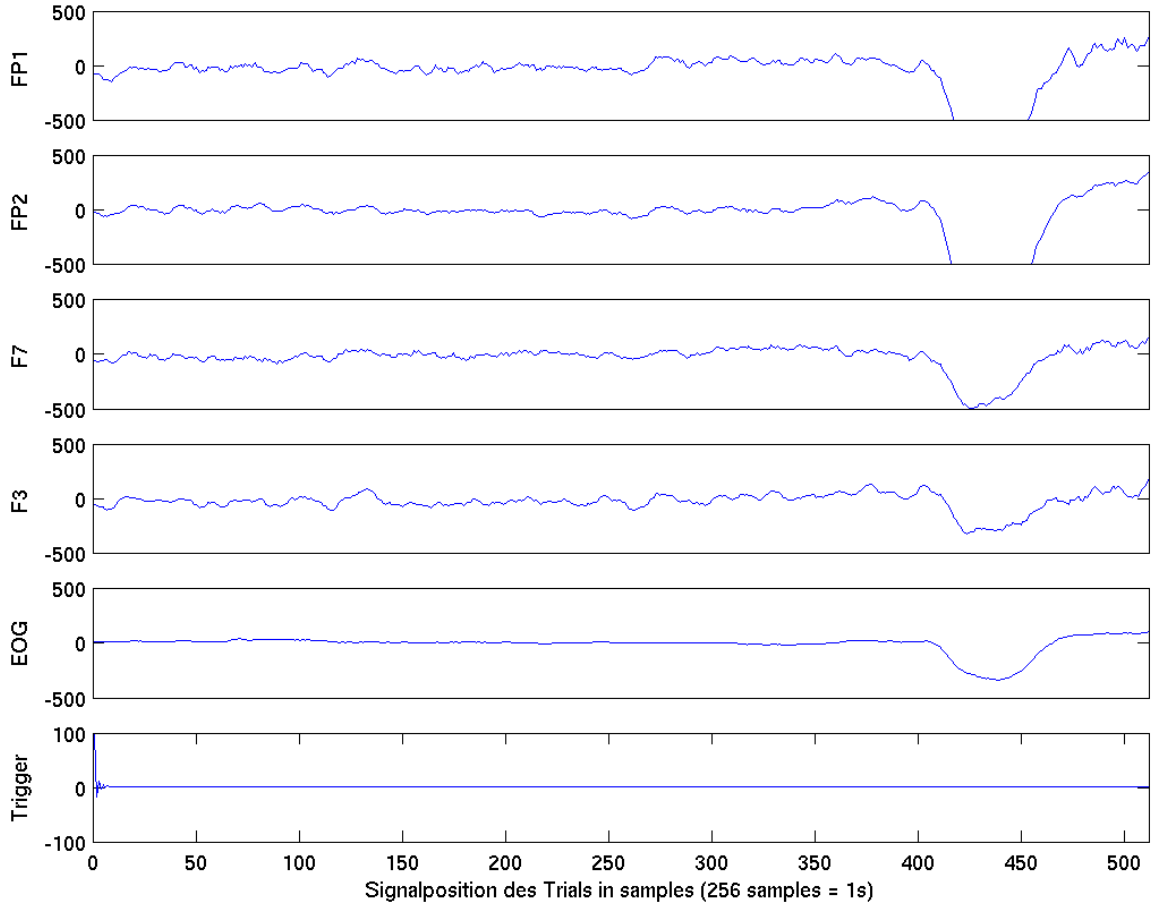

Abbildung 7.1.: Grafische Darstellung des aufgezeichneten EEG-Signals eines Trials (Vp 1, Bedingung *Abstrakt*). Abgebildet sind vier Signalelektroden (Fp1, Fp2, F7, F3) sowie der Augenkanal (EOG) und das Triggersignal. Am Ende des Trials ist sehr deutlich ein Artefakt durch Muskelaktivitäten zu sehen.

Die Merkmalsextraktion umfasste zwecks direkter Vergleichbarkeit mit den Untersuchungen in [WR96] nur die Bedingungen *abstrakte Nomen* (z.B. „Anlass “, „Begriff“, „Zweifel“) und *konkrete Nomen* (z.B. „Anker“, „Becher “, „Zwiebel“), was bei 17 Vp mit jeweils 2 Teilversuchen und 24 Präsentation eines Wortes (*Trials*) pro Teilversuch insgesamt 816 Sequenzen pro Bedingung ergibt.

Auf die jeweils 2 Sekunden langen Trials haben wir eine Short-Time-Fourier-Transformation (STFT, vgl. [SD96]) mit verschiedenen Fensterlängen (256, 128 und 64 samples) und -überlappungen (*Overlaps*) angewendet. Kleine Fensterbreiten ermöglichen dabei eine hohe Zeitauflösung an den Enden der Trials – z.B. liegt die Fenstermitte bei 64 samples bei 62.5 ms ab Präsentation des Wortes –, große dagegen eine genauere Frequenzauflösung (1Hz bei 256 samples gegenüber 4 Hz bei 64 samples). Als Fensterfunktion

wählten wir das *Hanning-Fenster*

$$h(k) = \frac{1}{2}\left(1 - \cos(2\pi k/n)\right), \tag{7.1}$$

wobei n die Länge des Fensters ist. Außerdem haben wir verschiedene Frequenzbänder und Elektrodenkonstellationen berücksichtigt. Die so entstandenen Spektrogramme (s. Abb. 7.2) bilden unsere Merkmalssequenzen.

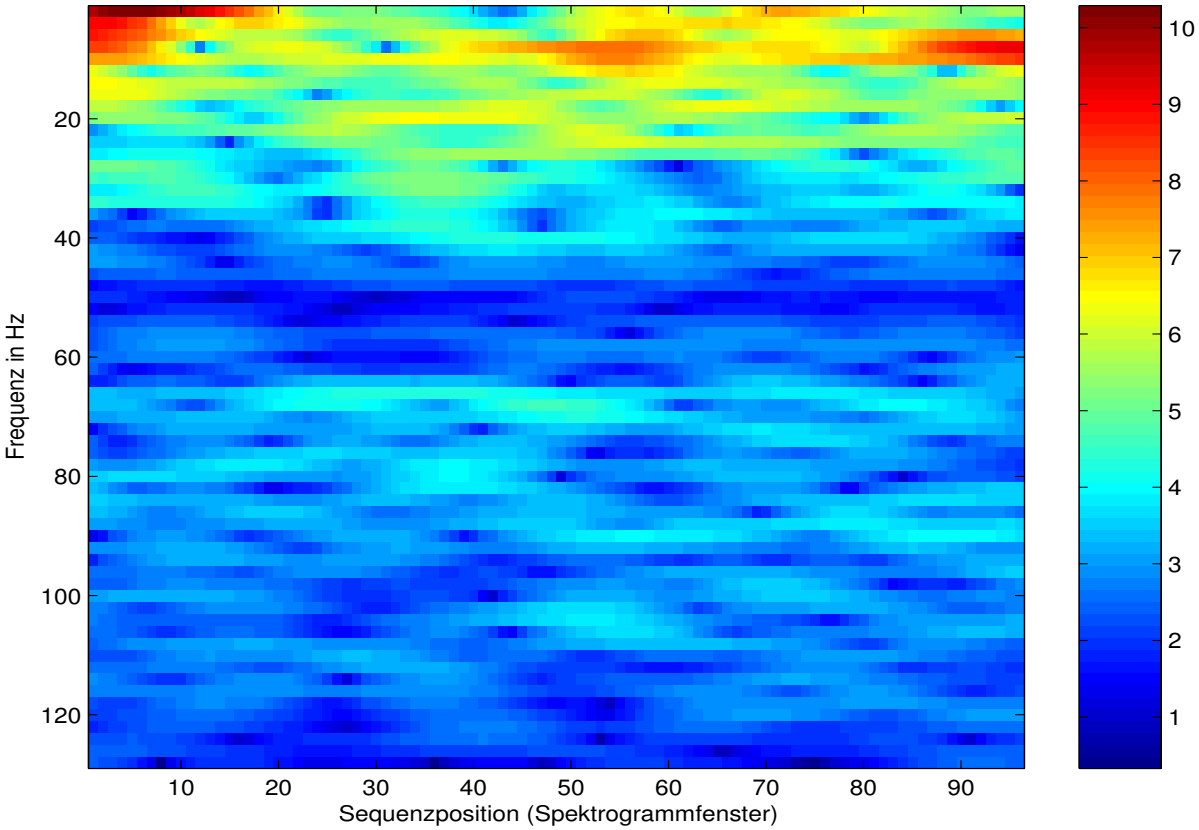

Abbildung 7.2.: Logarithmisch skaliertes Spektrogramm eines Trials von Versuchsperson 6 und Elektrode Fp1. Die Fensterbreite beträgt 128 samples, der Überlapp 124 samples. Präsentiert wurde in diesem Fall ein abstraktes Nomen. Der Werteabfall bei 50 Hz ist auf den Netzfrequenzfilter zurückzuführen.

Die Länge der Sequenzen hängt von der Fensterlänge und dem Overlap, die Dimensionalität von den verwendeten Elektroden und Frequenzbändern ab. Da die Sequenzen für fixierte Parameter gleich lang sind, ist der direkte Vergleich zwischen den OMMs (s. Kapitel 5), FAMs (s. Kapitel 6) und Standardmethoden (z.B. SVM, s. Abschnitt 2.7) möglich.

Bereinigung

Zu Vergleichszwecken haben wir bei der Merkmalsextraktion anhand des Augenkanals (EOG) Sequenzen mit einem Minimum unter dem Schwellwert von -100 μV aussortiert. Dies stellt eine einfache Art der *Artefaktelimination* dar. Wie viele Sequenzen pro Versuchsperson entfernt wurden, ist Tabelle 7.2 zu entnehmen.

VP	1	2	3	6	7	9	11	12	13	14	15	16	17	18	20	24	25
abstrakt	2	7	0	2	1	3	5	7	0	3	1	4	6	3	4	4	0
konkret	4	6	0	1	0	1	5	6	1	3	5	3	4	3	0	1	1

Tabelle 7.2.: Anzahl der pro VP aufgrund von Artefakten entfernten Trials (s. Text).

7.1.2. Protein-Daten

Proteine sind einer der Grundbausteine des Lebens. Ihre Bedeutung für biologische Organismen ist enorm, da sie im Körper und in den Muskeln mechanische Stütz- und Bewegungsfunktionen übernehmen und als *Hormone* eine Vielzahl von Vorgängen des Körpers steuern und regeln. Sie haben *enzymatische* Wirkungen, d.h. sie fungieren als Biokatalysatoren, übernehmen Transportfunktionen, Aufgaben in der Immunabwehr und übertragen Nervenimpulse.

Chemisch betrachtet bestehen Proteine aus *Aminosäuren*, die durch *Peptidbindungen* zu Ketten verbunden sind, daher spricht man auch von *Polypeptidketten*. Es gibt über 600 verschiedene Aminosäuren, von denen aber nur 20 proteinogen sind. Ein einzelnes Protein kann aus 20 bis über 1000 Aminosäuren bestehen.

Den Aminsosäuren wird anhand ihres Namens eine Abkürzung zugewiesen (beispielsweise *Alanin*, Abkürzung: `a`). Ein Wort eines 20-zeichigen Alphabets repräsentiert dann ein Protein (vgl. Abb. 7.3).

```
stkgpsvfplapsskstsggtaalgc...
```

Abbildung 7.3.: Der Anfang eines Proteins des von uns verwendeten SCOPSUPER95_66-Datensatzes.

Proteine sind also Sequenzen von Aminosäuren und daher zur Evaluation unserer Algorithmen geeignet. Die besondere diskrete Struktur aus nur 20 Bausteinen unterscheidet sie stark von den EEG-Daten, daher repräsentieren sie eine andere Sequenzart. Wir haben uns für die Proteinanalyse entschieden, da die Proteine einen sinnvollen, diskreten Gegenpol zu den kontinuierlichen EEG-Daten darstellen. Darüberhinaus werden Techniken des maschinellen Lernens noch verhältnismäßig selten zur Lösung bioinformatischer Probleme eingesetzt, versprechen aber gute Ergebnisse von denen sowohl Molekurlarbiologen als auch Bioinformatiker profitieren können.

Ursprung

Der von uns benutzte Proteindatensatz basiert auf dem etablierten SCOP-Klassifikationssystem (*Structural Classification of Proteins*, s. [MBH95], [LBH02], [AHB04]). In diesem Datensatz werden alle Proteine bekannter Struktur nach ihren evolutionären, funktionalen und strukturellen Beziehungen zusammengefasst und hierarchisch sortiert. Als Klassifikationsgrundlage dienen die *Proteindomänen*, die kleinsten Einheiten eines Proteins, die eine unabhängige, gefaltete Struktur besitzen. Proteine werden in SCOP hierarchisch in Familien (*Families*), Superfamilien (*Superfamilies*), Folds und Klassen nach unterschiedlichen Kriterien zusammengefasst:

Family: Proteine werden nach zwei Anhaltspunkten in Familien eingeordnet. Zum einen umfasst eine Familie alle Proteine, die eine hohe Sequenzähnlichkeit auf Aminosäureebene aufweisen, zum anderen alle Proteine, deren Funktion und Struktur *sehr* ähnlich sind, beispielsweise die *Globine*.

Superfamily: In Superfamilien werden die Familien zusammengefasst, die zwar eine u. U. geringere Sequenzähnlichkeit haben, allerdings von ihren strukturellen und funktionalen Eigenschaften so ähnlich sind, dass ein gemeinsamer evolutionärer Ursprung wahrscheinlich ist.

Fold: In gemeinsamen Folds werden die Superfamilies organisiert, deren Proteine die gleichen *Sekundärstrukturen*, also die gleichen räumlichen Anordnungen haben.

Class: Auf höchster Hierarchieebene werden die Folds in Klassen nach Strukturtypen zusammengefasst, beispielsweise bilden alle *alpha*-Proteine eine Klasse. Insgesamt umfasst SCOP aktuell 11 Klassen.

Die von uns benutzte Teilmenge SCOPSUPER95_66 des SCOP-Datensatzes basiert auf der SUPERFAMILY-Hierarchie von Julian Gough (s. [Gou01]) und wurde von Thomas Plötz und Gernot A. Fink zusammengestellt (vgl. [PF04]). Der Name basiert auf der Tatsache, dass auf Ebene der Superfamilien klassifiziert wird, zusammengehörige Superfamilien eine Ähnlichkeit von maximal 95% haben dürfen (vgl. Abb. 7.4) und jede Klasse in Trainings- und Testmenge mindestens 66 Sequenzen umfasst.

Zusätzlich zu den 20 Aminosäuren gibt es einen Platzhalter `x`, der dann eingfügt wird, wenn unklar ist, welche Aminosäure sich an dieser Stelle befindet.

Der Datensatz beinhaltet 16 Klassen mit insgesamt 1120 Trainingssequenzen. Zusätzlich steht eine disjunkte Testmenge von 566 Testsequenzen zur Verfügung. Die Varianz der Sequenzlänge ist sehr groß. Die längste Sequenz ist 795 Aminosäuren lang, die kürzeste 22. In Tabelle 7.3 werden diese Informationen genauer dargestellt.

Transformation

Um die Proteine in multivariate Sequenzen zu übersetzen, verfolgen wir zwei Ansätze: Indexvektoren und biologisch motivierte Blosumvektoren.

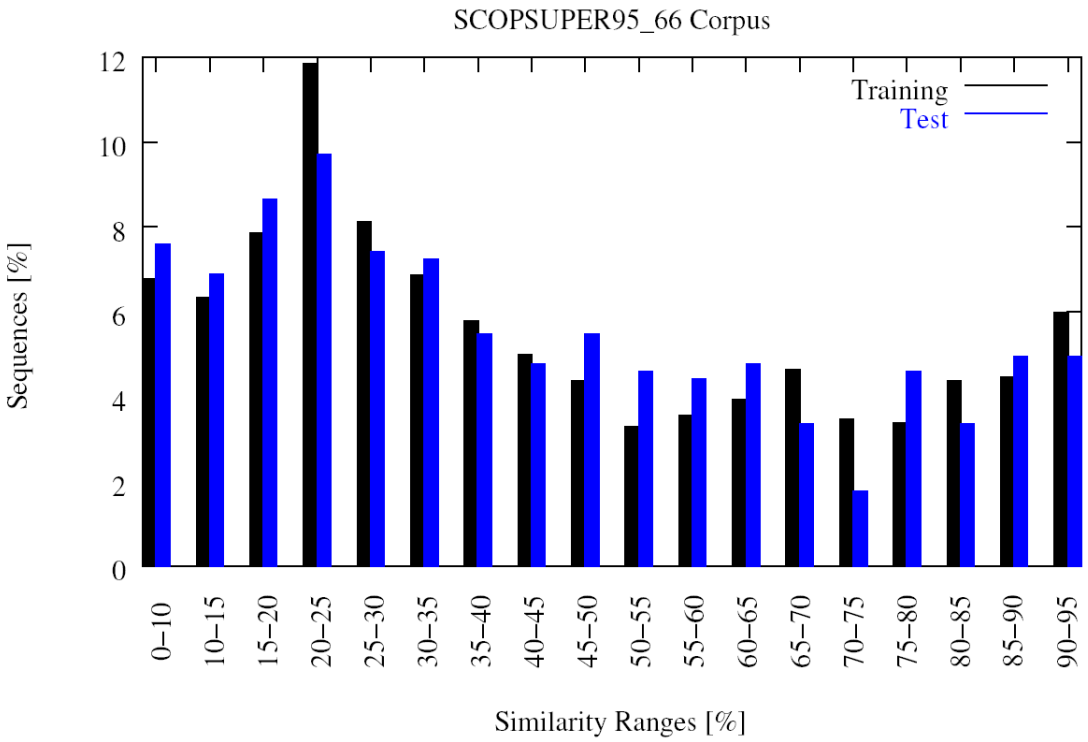

Abbildung 7.4.: Prozentualer Anteil von Sequenzen des SCOPSUPER95_66 Datensatzes nach Sequenzähnlichkeit (aus [Plö05]).

Indexvektoren ersetzen jeweils eine Aminosäure durch einen Indexvektor $\mathbf{s} \in \{0,1\}^{20}$. Jeder Aminosäure wird einer von 20 Indexvektoren zugeordnet, Proteine werden dann als Folge dieser Vektoren (also als Matrizen) repräsentiert und können so von unseren Algorithmen verarbeitet werden. In Abbildung 7.5 ist der Anfang einer solchen Matrix dargestellt.

Die Indexvektor-Repräsentation unterscheidet nicht zwischen Aminosäuren. Es gibt Aminosäurenpaare, die sich in ihren chemischen Eigenschaften und Strukturen ähneln und daher größtenteils ohne Einfluss auf die Funktion des Proteins austauschbar sind. Indexvektoren vernachlässigen diese Information, die Distanz zwischen zwei verschiedenen Aminosäuren ist in dieser Repräsentation immer gleich.

Blosumvektoren sind biologisch motiviert. Eine 20×20-dimensionale Blosummatrix $\mathbf{B}$ beschreibt die Austauschwahrscheinlichkeiten zwischen den Aminosäuren. An Position b_{ij} der Matrix steht die logarithmierte Austauschwahrscheinlichkeit zwischen der i-ten und der j-ten Aminosäure (vgl. Abschnitt 3.2). Aus diesen Matrizen lassen sich nun, ähnlich den Indexvektoren, 20-dimensionale Merkmalsvektoren für die Aminosäuren extrahieren, die j-te Aminosäure wird durch den j-ten Spaltenvektor aus der Blosummatrix ersetzt. Wir verwenden die Blosum62-Matrix (s. Abb. 7.6).

Mit diesen Vektoren lassen sich nun zwei Aminosäuren bezüglich eines Ähnlichkeitsmaßes (beispielsweise der negativen Distanz oder des Skalarprodukts) in Relation zueinander setzten.

```
0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  1  0  0  0  0
0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  1  1  0  0  1  0  0  0  0  1  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  1  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0
1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0
0  0  0  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ...

s  t  k  g  p  s  v  f  p  l  a  p  s  s  k  s  t  s  g  g  t  a  a  l  g  c ...
```

Abbildung 7.5.: Der Anfang des Proteins aus Abbildung 7.3 in Indexvektordarstellung.

```
-2  4 -1  1 -1  1 -1  0  0 -1 -1 -1 -1 -1 -1 -1  4 -1  0 -1  0  4 -2 -1 -1  0
-3 -1  5 -1 -2 -1 -3 -1 -1 -2  0  0 -1 -3  0  2 -1  2 -2  0 -1 -1 -2 -2 -2 -3
-3 -2  0  1 -3  1 -3  0  0 -2  0  0 -2 -3  0  0 -2  0  0  0  0 -2 -2 -3 -2 -3
-3 -2 -2  0 -4  0 -3 -1 -1 -1  2  2 -3 -3  2 -1 -2 -1 -1  2 -1 -2 -3 -4 -1 -3
-2  0 -3 -1 -1 -1 -1 -1 -1 -3 -4 -4 -1 -1 -4 -3  0 -3 -3 -4 -1  0 -2 -1 -3  9
-3 -1  1  0 -2  0 -3 -1 -1 -1  2  2  0 -3  2  1 -1  1 -2  2 -1 -1 -1 -2 -1 -3
-3 -1  0  0 -3  0 -3 -1 -1 -1  5  5 -2 -3  5  1 -1  1 -2  5 -1 -1 -2 -3 -1 -4
-3  0 -2  0 -4  0 -4 -2 -2 -2 -2 -2 -3 -4 -2 -2  0 -2  6 -2 -2  0 -3 -4 -2 -3
-1 -2  0 -1 -3 -1 -3 -2 -2 -2  0  0 -2 -3  0 -1 -2 -1 -2  0 -2 -2  2 -3 -2 -3
 0 -1 -3 -2  2 -2  4 -1 -1 -3 -3 -3  1  4 -3 -3 -1 -3 -4 -3 -1 -1 -1  2 -3 -1
 0 -1 -2 -2  4 -2  2 -1 -1 -3 -3 -3  2  2 -3 -2 -1 -2 -4 -3 -1 -1 -1  4 -3 -1
-3 -1  2  0 -2  0 -3 -1 -1 -1  1  1 -1 -3  1  5 -1  5 -2  1 -1 -1 -2 -2 -1 -3
 0 -1 -1 -1  2 -1  1 -1 -1 -2 -2 -2  5  1 -2 -1 -1 -1 -3 -2 -1 -1 -1  2 -2 -1
 6 -2 -3 -2  0 -2  0 -2 -2 -4 -3 -3  0  0 -3 -3 -2 -3 -3 -3 -2 -2  3  0 -4 -2
-4 -1 -2 -1 -3 -1 -3 -1 -1  7 -1 -1 -2 -3 -1 -1 -1 -1 -2 -1 -1 -1 -3 -3  7 -3
-2  1 -1  4 -2  4 -2  1  1 -1  0  0 -1 -2  0  0  1  0  0  0  1  1 -2 -2 -1 -1
-2  0 -1  1 -1  1 -1  5  5 -1 -1 -1 -1 -1 -1 -1  0 -1 -2 -1  5  0 -2 -1 -1 -1
 1 -3 -3 -3 -2 -3 -3 -2 -2 -4 -3 -3 -1 -3 -3 -3 -3 -3 -2 -3 -2 -3  2 -2 -4 -2
 3 -2 -2 -2 -1 -2 -1 -2 -2 -3 -2 -2 -1 -1 -2 -2 -2 -2 -3 -2 -2 -2  7 -1 -3 -2
-1  0 -3 -2  1 -2  3  0  0 -2 -2 -2  1  3 -2 -2  0 -2 -3 -2  0  0 -1  1 -2 -1 ...

 s  t  k  g  p  s  v  f  p  l  a  p  s  s  k  s  t  s  g  g  t  a  a  l  g  c ...
```

Abbildung 7.6.: Der Anfang des Proteins aus Abbildung 7.3 in Blosumvektordarstellung.

SCOPid	SCOP Superfamily Name	#Beispiele		Länge (Mean/Std.abweichung)	
		Training	Test	Training	Test
a.1.1	Globin-like	60	30	150.3 (13.6)	151.6 (11.1)
a.3.1	Cytochrome c	44	22	102.6 (24.1)	118.4 (32.6)
a.39.1	EF-hand	49	25	138.1 (48.0)	122.0 (39.3)
a.4.5	"Winged helix" DNA-binding domain	49	25	93.8 (26.6)	92.9 (23.1)
b.1.1	Immunoglobulin	207	104	108.9 (15.3)	106.7 (12.3)
b.10.1	Viral coat and capsid proteins	64	32	278.0 (92.9)	262.1(85.2)
b.29.1	Concanavalin A-like lectins/glucanases	52	27	221.2 (51.2)	220.8 (72.9)
b.40.4	Nucleic acid-binding proteins	47	24	113.1 (36.6)	111.5(47.2)
b.47.1	Trypsin-like serine proteases	55	28	231.4 (29.5)	226.0 (30.1)
b.6.1	Cupredoxins	50	26	143.9 (34.6)	139.0 (31.5)
c.1.8	(Trans)glycosidases	62	31	376.5 (76.4)	397.8 (84.0)
c.2.1	NAD(P)-binding Rossmann-fold domains	102	51	204.3 (58.9)	211.5 (75.1)
c.3.1	FAD/NAD(P)-binding domain	45	23	226.1 (93.3)	223.3 (86.3)
c.37.1	P-loop containing nucleotide triphosphate hydrolases	127	64	259.3 (120.4)	253.4 (85.6)
c.47.1	Thioredoxin-like	56	28	111.6 (38.2)	105.6 (35.3)
c.69.1	Alpha/Beta-Hydrolases	51	26	350.1 (103.7)	323.7 (25.0)

Tabelle 7.3.: Abgebildet sind Informationen über den verwendeten SCOPSUPER95_66-Datensatz (nach [Plö05]).

Behandlung der Platzhalter

Unklar bei dem oben beschriebenen Vorgehen bleibt die Behandlung der Platzhalter für die nicht-eindeutigen Aminosäuren `x`. Denkbar wäre beispielsweise, diese aus den Proteinen zu entfernen. Allerdings repräsentieren sie eine Aminosäure, wenn auch unklar ist welche, und sind daher für die biologischen Eigenschaften des entsprechenden Proteins bedeutend.

Wir haben uns entschieden, die Platzhalter durch Aminosäuren-Merkmalsvektor-Mittelwerte zu ersetzen. Sie bleiben als uneindeutige Sequenzmerkmale im Datensatz erhalten und zeigen auf diese Weise ihre Natur: die Belegung mit allen Aminosäuren ist möglich, nur ist eben nicht klar, um welche es sich handelt.

7.2. Experimente

Ausgehend von den eingangs beschriebenen Datensätzen bieten sich uns vielfältige Untersuchungsmöglichkeiten. Da nicht jede Kombination von Methoden (z.B. *FAMmean*, *OMMbest*), Varianten zur Merkmalsgewinnung (z.B. Spektrogramme, Indexvektoren), Hyperparametern (evaluierbare Parameter der Methoden wie z.B. die Prototypgröße K) und Testumgebung (z.B. Anzahl der Folds in der Kreuzvalidierung) sinnvoll und zeitlich realisierbar ist, haben wir uns auf die wichtigsten und interessantesten beschränkt.

Dabei sind wir hierarchisch vorgegangen, d.h. zuerst haben wir die jeweiligen Daten

grob analysiert um dann vielversprechende bzw. auffällige Ergebnisse genauer zu untersuchen. Wir werden in diesem Abschnitt lediglich die Experimente, nicht jedoch die Ergebnisse – die unter Umständen den Ausschlag zu diesen gegeben haben – beschreiben.

7.2.1. EEG-Daten

Bei diesen Daten war das Ziel, die anhand von EEG-Aufzeichungen gewonnen Reaktionen auf unterschiedliche Wortarten unterscheiden zu können und neu präsentierte Wörter richtig einzuordnen. Das 2-Klassen-Problem besteht dabei aus einer Menge von *abstrakten* Wörtern (Klasse 1) sowie einer Menge von *konkreten* Wörtern (Klasse 2).

Alle Versuche fanden zunächst auf einzelnen Versuchspersonen statt, d.h. die Aufgabe bestand darin, die Aktivitätsmuster einzelner Personen zu lernen. Später führten wir dann auch Generalisierungstests über die Vps hinweg durch.

Bereits die Merkmalsextraktion der EEG-Daten bietet eine Vielzahl von Parametern bzw. Optionen. So beinhaltet die Spektrogrammberechnung folgende Möglichkeiten:

- **Fensterfunktion und -breite:** Hier haben wir uns auf lediglich eine Fensterfunktion (s. Abschnitt 7.1) mit drei unterschiedlichen Fensterbreiten ($n = 256$, $n = 128$ und $n = 64$ samples) beschränkt. Welche Auswirkungen die verwendeten Fensterbreiten n auf die Frequenzauflösung und den Startpunkt der Untersuchung ab der Präsentation des Wortes haben, ist in Tabelle 7.4 zusammengefasst. Aufgrund der an der Enden stark abfallenden Hanning-Fensterfunktion (s. Gl. 7.1) werden dabei Signalwerte des ersten Quantils des ersten Fensters und die des letzten Quantils des letzten Fensters nur ungenügend berücksichtigt.

- **Fensterüberlappung:** Um eine hohe Zeitauflösung der Spektrogramme zu erreichen, haben wir den Overlap so gewählt, dass diese unter den von H. Müller empfohlenen 20 ms liegt. Bei einer Abtastrate (*sampling rate*) der Daten von 256 Hz entspricht dies $n - 5$ samples. Aus Berechnungsgründen wählten wir jedoch $n - 4$ samples. Die sich daraus ergebende Zeitauflösung und Anzahl der Fenster pro Sequenz lässt sich ebenfalls in Tabelle 7.4 ablesen.

n	o	R_t	R_f	Beginn	Ende	# Fenster
256	252	15.6 ms	1 Hz	250 ms	1750 ms	64
128	124	15.6 ms	2 Hz	125 ms	1875 ms	96
64	60	15.6 ms	4 Hz	63 ms	1937 ms	112

Tabelle 7.4.: Fensterbreiten n und davon abhängiger Overlap o, Zeit- und Frequenzauflösung (R_t bzw. R_f), Beginn und Ende der genauen Untersuchung (s. Text) sowie Anzahl der entstehenden Fenster.

Anschließend lässt sich zudem der **Frequenzbereich** durch Ausblenden der nicht benötigten Koeffizienten eingrenzen. Hier haben wir uns zunächst dem in [WR96] anhand von Kohärenzberechnungen untersuchten Bereich von 13-18 Hz – dem β_1-Band –

gewidmet. Da in dieser Arbeit nur das α_1- (8-10Hz) und das β_1-Band analysiert, viele kognitive Prozesse jedoch in höheren Frequenzen vermutet werden (vgl. [Zsch95]), dehnten wir die Versuche auf das β_2-Band (18-30 Hz) und das γ-Band (über 30 Hz) aus. Außerdem untersuchten wir kleinere Frequenzbänder im gesamten nutzbaren Spektrum von 1-120 Hz.

Durch Auswahl der zur Untersuchung herangezogenen **Elektroden** lassen sich die Aktivitäten einzelner Areale genauer betrachten und somit spezifische kognitive Komponenten lokalisieren. Zu diesem Zweck führten wir eine Analyse für jede einzelne der insgesamt 19 Signalelektroden durch – auch um anschließend die besten Elektroden zu kombinieren und somit evtl. vorhandene störende Elemente zu eliminieren.

Weitere Zusammenfassungen von Elektroden ergaben folgende Areale:

- **Front:** Fp1, Fp2,
- **links anterior:** Fp1, F7, T3,
- **links posterior:** T5, O1,
- **rechts anterior:** Fp2, F8, T4,
- **rechts posterior:** T6, O2,
- **Hinten:** O1, O2.

Schließlich prüften wir auch die Auswirkung von **Artefakten** auf die Ergebnisse, indem wir (wie in Abschnitt 7.1 beschrieben) die artfaktbehafteten Trials entfernten und die Experimente mit diesen Daten wiederholten.

Während wir die Konfiguration mit allen Elektroden und verschiedenen Fensterlängen mit allen **Methoden** untersuchten, beschränkten wir uns für die spezifischen Analysen auf die vielversprechendste Methode *OMMbest*.

7.2.2. Proteindaten

Ziel der Analysen der Proteindaten war zunächst, die grundsätzliche Eignung unserer **Algorithmen** für diese Domäne zu untersuchen. Dazu führten wir detaillierte Experimente mit Ordered-Means-Modellen und Feature-Alignment-Maschinen durch. Einen besonderen Schwerpunkt legten wir dabei auf Klassifikationsanalysen, denn dieses Problemfeld ist seit Langem im Interesse bioinformatischer Untersuchungen und ermöglicht eine Vergleichbarkeit unserer Algorithmen mit etablierten Verfahren.

Nicht weniger interessant, wenn auch weniger verbreitet, ist eine **Hauptkomponentenanalyse** auf Proteinen, die längerfristig weitere biologische Erkenntnisse verspricht. Wir begrenzen uns auf reine Machbarkeitsstudien, denen weitere Analysen folgen müssen. Aufgrund der Notwendigkeit einzelne Aspekte unserer Algorithmen zu fokussieren, konnten diese von uns im Rahmen dieser Diplomarbeit nicht durchgeführt werden.

Darüberhinaus versuchten wir zu klären, ob zusätzliches biologisches Wissen in Form komplexerer **Merkmale** die Aufgabe der Sequenzklassifikation erleichtert. Die Vergleiche zwischen Index- und Blosumvektoren können hier Hinweise liefern.

Zuletzt stellt sich die Frage, inwieweit die gelernten **Prototypen** interpretierbar sind und ob sie Eigenschaften einer Proteinklasse widerspiegeln.

7.2.3. Parameter der eingesetzten Verfahren

Neben den eigentlichen Parametern eines Verfahrens, die durch den Lernprozess bestimmt werden – z.B. die Mittelwerte der Emissionsdichten bei *OMMall* –, gibt es meist weitere feste Parameter, die vor dem Training festgelegt werden. Diese sogenannten *Hyperparameter* sind Bestandteil der Modellselektion und mittels der in Abschnitt 2.9 beschriebenen Kreuzvalidierung evaluierbar.

SVM

Die zum Vergleich auf den EEG-Daten verwendete C-SVM besitzt zwei Hyperparameter. Zum einen ist dies die Kernfunktion mit dem Kernparameter – in diesem Fall der Gausskern mit der Kernbreite γ –, zum anderen der Faktor C des Strafterms.

OMM

OMMbest Die unregularisierte Variante dieses Verfahrens besitzt lediglich die (diskrete) Prototypgröße K als Hyperparameter. Im regularisierten Fall kommt der Faktor $\lambda \in [0, 1]$ für den Strafterm hinzu.

OMMall Zusätzlich zur Prototypgröße K ist hier der Varianzparameter der Emissionsdichten σ einzustellen. Dabei ist zu beachten, dass σ nicht zu niedrig angesetzt wird, da sich sonst die numerischen Ungenauigkeiten negativ auf das Konvergenzverhalten des Trainings auswirken.

OMMKDE Da hier lediglich zwei Sequenzen gegeneinander aligniert werden, steht der Parameter K als Länge der zweiten Sequenz fest. Jedoch ist der Einfluss der Kernbreite σ von großer Bedeutung und muss daher sehr fein abgetastet werden.

FAM

Allen FAM-Methoden ist der Hyperparameter D gemein, welcher das Produkt aus Folgenlänge L und Prototypgröße K ist. Die Liste der Folgenlängen L ist ebenfalls manuell festzulegen, jedoch kein wirklicher Hyperparameter, da nur wenige sinnvolle Werte existieren, die zudem vor dem Training anwendungsabhängig ermittelbar sind. Dazu kann man einzelne Werte für ein Teilproblem evaluieren und die besten kombinieren.

FAMmean Der Mittelwertschätzer im FAM-Merkmalsraum beschränkt sich ausschließlich auf den Hyperparameter D .

FASVM Die Ein-Klassen-Variante der *FASVM* verwendet in der Regression eine ν-SVM. Daher muss zusätzlich zu D noch die Gewichtung ν des Strafterms evaluiert werden.

diskriminative FASVM Die diskriminative Variante der *FASVM* benutzt ebenfalls eine SVM Variante im Regressionsschritt. Auch hier muss eine geeignete Gewichtung C des Strafterms bestimmt werden.

8. Ergebnisse

8.1. EEG-Daten

Für diese Domäne stellte sich schnell heraus, dass die *OMMs* leistungsfähige und interpretierbare Prototypen erzeugen. Die Abbildungen 8.1 und 8.2 zeigen z.B. den gut unterscheidbaren Verlauf der diskreten Zuordnungen bzw. Verantwortlichkeiten für die verschiedenen Kategorien.

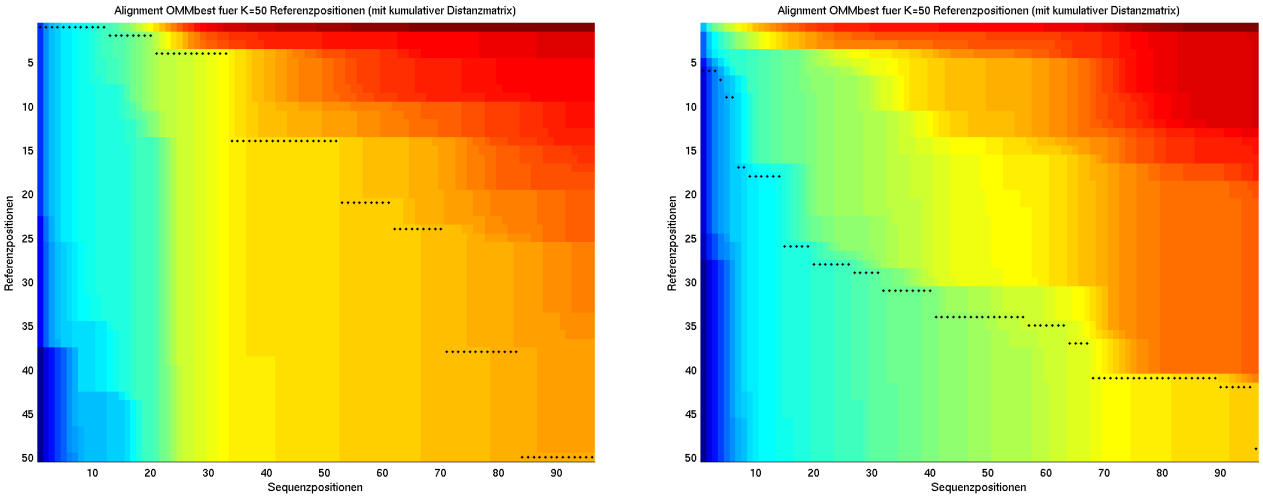

Abbildung 8.1.: Beispielhafte Visualisierungen des Alignments von *OMMbest* für ein abstraktes Wort (links) und ein konkretes Wort (rechts). Die unterschiedlichen Alignmentpfade lassen auf unterschiedliche zeitliche Relevanz der Sequenzabschnitte schließen. Die Prototypgröße (K) beträgt in diesem Fall 50, der Regularisierungsparameter (λ) 0.5.

Jedoch auch die *FAMs* konnten im Frequenzbereich von 13-18 Hz überzeugen. Zwar sind die grafischen Umsetzungen des Alignments (s. Abb. 8.4) nicht so deutlich verschieden wie die der *OMMs*, trotzdem kann man leichte qualitative und quantitative Unterschiede ausfindig machen.

Obwohl wir die Wahrscheinlichkeitswerte konsequent logarithmisch berechnet haben, kam es bei der Wahl sehr kleiner σ bei *OMMall* zu degenerierten Alignments (deterministische Zuordnungnen auf einen Prototypen, s. Abb. 8.3). Dies wirkte sich auf den Wert der Fehlerfunktion aus, sodass auch diese nicht mehr monoton konvergierte. Ursache

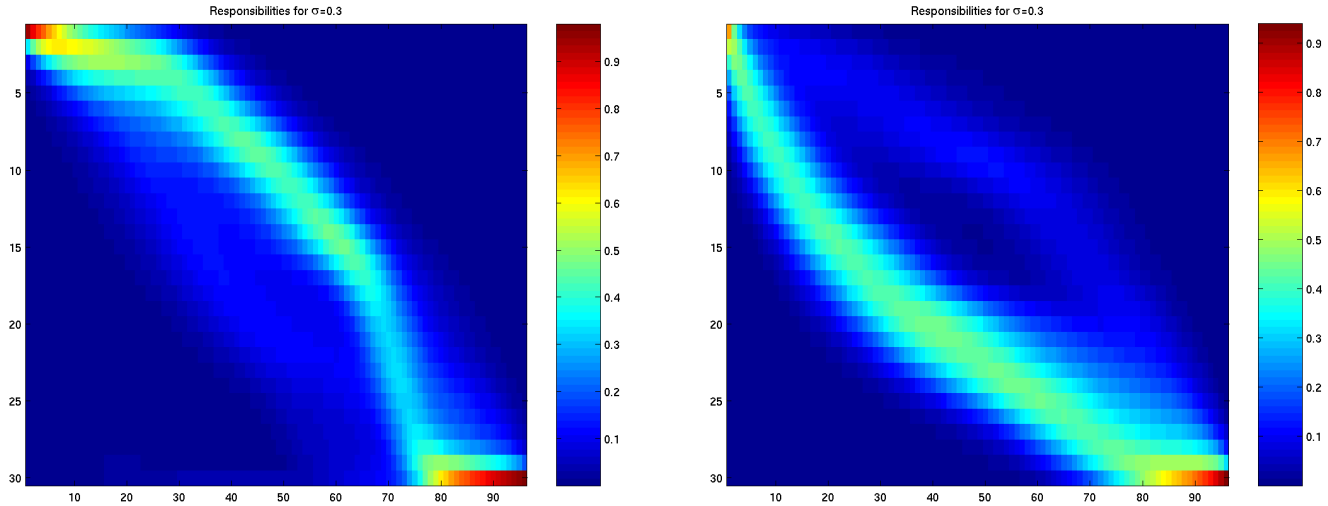

Abbildung 8.2.: Beispielhafte Visualisierungen des Alignments von *OMMall* für ein abstraktes Wort (links) und ein konkretes Wort (rechts). Wieder zeigt sich die Auswirkungen der bedeutenden zeitlichen Informationen. Zusätzlich kann beim konkreten Wort ein schwacher Alternativpfad erkannt werden.

dafür ist die numerische Rechengenauigkeit der PCs, die mit 64 Bit für größe K bzw. T unzureichend ausfällt.

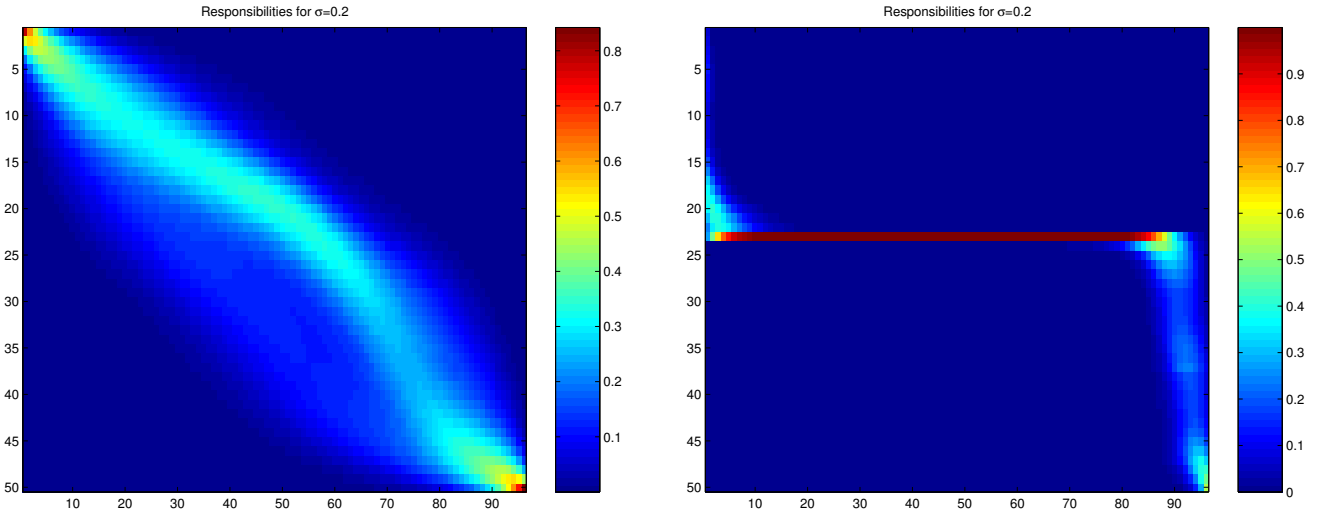

Abbildung 8.3.: Degeneriertes Alignment für ein abstraktes Wort mit *OMMall* aufgrund numerischer Rechenungenauigkeiten (s. Text). Links die Verantwortlichkeiten für die erste, rechts die für die 5. Iteration. Der Parameter σ beträgt in diesem Fall 0.2.

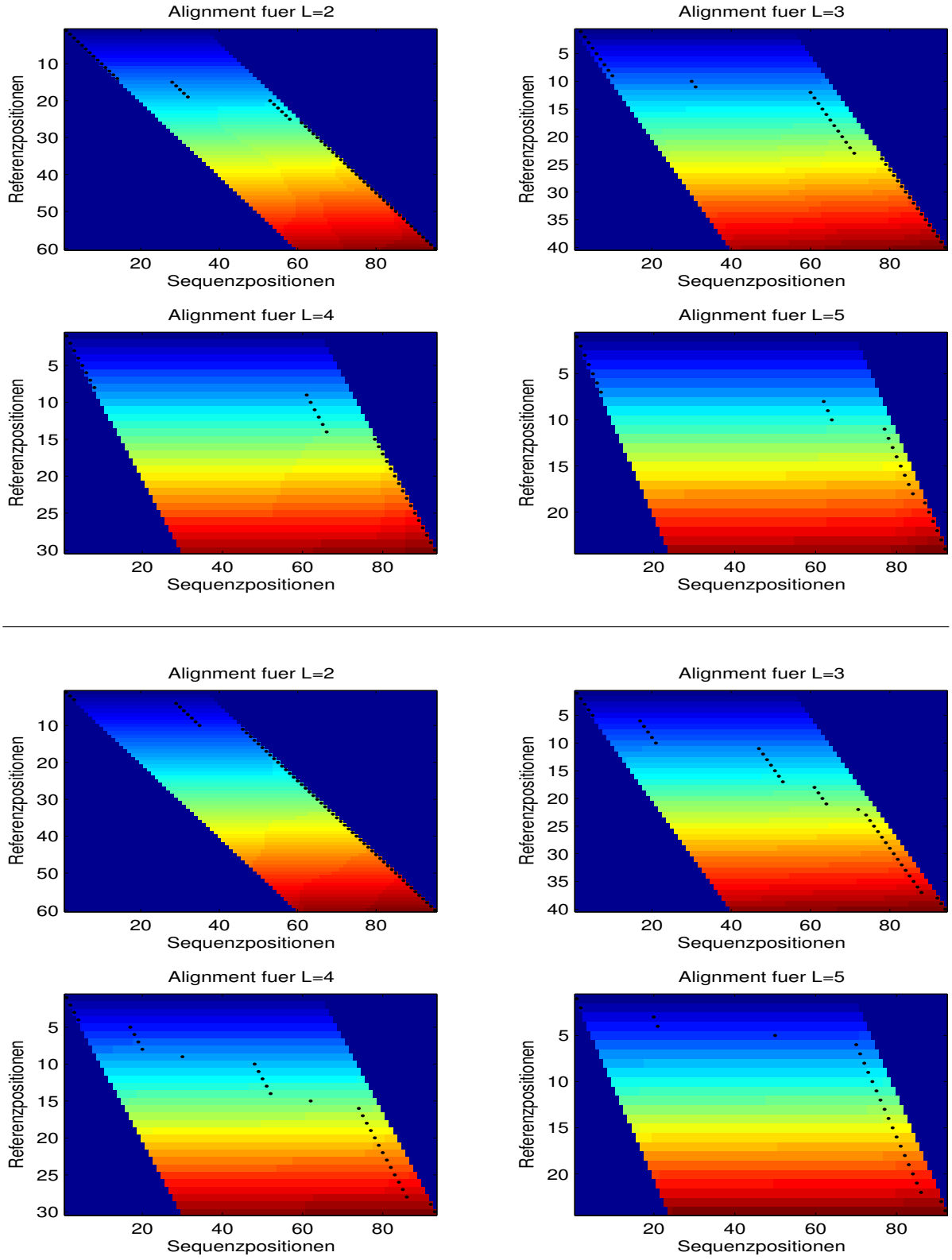

Abbildung 8.4.: Beispielhafte Visualisierungen des Alignments der *Feature-Alignment-Maschinen* für $L = \{2, 3, 4, 5\}$ für ein abstraktes Wort (obere 4 Bilder) und ein konkretes Wort (untere 4 Bilder).

8.1.1. Ergebnisse der Methoden im Überblick

Da die Sequenzen alle gleich lang sind, haben wir eine 2-Klassen-SVM (s. Abb 2.7) als Vergleichsmethode gewählt. Diese berücksichtigt jedoch die zeitliche Entwicklung des Signals im Gegensatz zu den alignmentbasierten Methoden nicht. Die Ergebnisse in den Tabellen 8.1 bis 8.3 stellen den Mittelwert der besten Ergebnisse aus der jeweiligen Kreuzvalidierung jeder Versuchsperson dar. Die Hyperparameter, welche für die besten Ergebnisse verantwortlich waren, sind aus Platzgründen nicht aufgeführt.

Fensterbreite	64	64a	128	128a	256	256a
SVM	67.03 %	67.56 %	**68.48 %**	68.01 %	68.20 %	63.29 %
OMMbest	76.04 %	76.28 %	75.75 %	**77.72 %**	75.36 %	76.91 %
OMMall	70.93 %	65.75 %	**72.38 %**	67.74 %	64.88 %	64.78 %
OMMKDE	68.70 %	68.16 %	69.67 %	69.14 %	67.49 %	**71.56 %**
FAMmean	60.06 %	59.98 %	**62.81 %**	60.73 %	61.05 %	62.06 %
FASVM	72.03 %	72.92 %	**76.10 %**	73.85 %	62.97 %	64.30 %
FASVMdisc	—	—	63.94 %	**64.45 %**	—	—

Tabelle 8.1.: Klassifikationsergebnisse für verschiedene Methoden auf EEG-Spektrogrammen **13-18 Hz**. Ein „a“ hinter der Fensterbreite bedeutet, dass artefaktbehaftete Trials entfernt wurden. Man sieht, dass fast alle von uns entwickelten Methoden die SVM auf kleinen Fensterbreiten übertreffen. *OMMbest* kann sich nur knapp gegen *FASVM* durchsetzen, wobei diese ohne die Artefaktbereinigung sogar etwas besser abschneidet. Aufgrund des sehr viel höheren Rechenaufwands haben wir bei der diskriminativen Variante der *FASVM* nur zwei Experimente gerechnet und die Ergebnisse vergleichshalber aufgenommen. Die schlechten Ergebnisse sind vermutlich auf den grob gewählten Hyperparameterumfang zurückzuführen. Die fett gedruckten Ergebnisse markieren die Reihenmaxima.

Während im β_1-Frequenzband (13-18 Hz) die optimale Fensterbreite bei 128 samples liegt (außer *OMMKDE*), beträgt sie im β_2-Band (18-30 Hz) für die OMM-Varianten 128 und für die FAM und SVM 256 samples. Im unteren γ-Band (30-40 Hz) hingegen setzen sich niedrige Fensterbreiten von 64 samples durch, was auf wichtige Komponenten am Beginn des Signals in diesem Frequenzband schließen lässt. Für $n = 256$ samples fällt die Erkennungsleistung für alle Methoden ab.

Die Entfernung artefaktbehafteter Trials bewirkt nur im β_1-Band und auch nur für *OMMbest* und *OMMKDE* eine Verbesserung der Erkennungsleistung. Aus diesem Grund haben wir im γ-Band auf die Evaluation dieses Parameters verzichtet.

Insgesamt fällt (besonders im β_2-Band) die Homogenität der Ergebnisse von *OMMbest* auf, wohingegen die anderen Methoden starke Schwankungen der Performanz in Abhängigkeit von der Fensterbreite und Entfernung von Trials aufweisen.

Fensterbreite	64	64a	128	128a	256	256a
SVM	79.62 %	74.72 %	82.72 %	77.52 %	**82.76 %**	74.66 %
OMMbest	86.85 %	85.39 %	**87.63 %**	84.74 %	84.26 %	81.66 %
OMMall	82.14 %	74.69 %	**83.99 %**	76.14 %	80.90 %	75.73 %
OMMKDE	74.17 %	75.44 %	**78.54 %**	76.71 %	75.39 %	76.69 %
FAMmean	69.18 %	65.67 %	68.11 %	66.12 %	**71.46 %**	67.59 %
FASVM	70.90 %	69.20 %	72.17 %	64.45 %	**73.68 %**	70.00 %
FASVMdisc	—	—	**73.39 %**	72.26 %	—	—

Tabelle 8.2.: Klassifikationsergebnisse für verschiedene Methoden auf EEG-Spektrogrammen **18-30 Hz**. Das beste Verfahren ist wiederum *OMMbest*, wobei der Vorsprung gegenüber der SVM weniger geworden ist. Am schlechtesten schneidet die *FASVM* ab. Die zwei vergleichshalber aufgenommenen Ergebnisse der diskriminativen *FASVM* reichen an die der Ein-Klassen-*FASVM* heran.

Fensterbreite	64 samples	128 samples	256 samples
SVM	**88.78 %**	88.25 %	83.96 %
OMMbest	**96.50 %**	95.05 %	90.71 %
OMMall	**93.57 %**	92.61 %	89.16 %
OMMKDE	85.07 %	**85.27 %**	82.73 %
FAMmean	76.12 %	**76.82 %**	75.56 %
FASVM	78.85 %	**79.45 %**	77.24 %
FASVMdisc	**78.88 %**	78.24 %	76.69 %

Tabelle 8.3.: Klassifikationsergebnisse für verschiedene Methoden auf EEG-Spektrogrammen **30-40 Hz**. Wiederum ist *OMMbest* das beste Verfahren, wobei der Vorsprung zur SVM wieder angestiegen ist. Dafür ist der Abstand von *FASVM* zu den anderen Methoden viel größer geworden. Alle Methoden schneiden für dieses Frequenzband (aus den drei untersuchten) am besten ab. Im Gegensatz zu den anderen Frequenzbändern haben wir hier auch die rechenaufwändige diskriminative Variante der *FASVM* evaluiert. Die Ergebnisse dieser liegen im Bereich der Ein-Klassen-Variante.

8.1.2. Untersuchung nach Frequenzen

Für die Untersuchung nach wichtigen Frequenzen extrahierten wir nur jeweils einen Frequenzkoeffizienten pro Elektrode und erhielten somit für eine Fensterbreite von 128 samples ein Frequenzband von 2 Hz. Beispielhaft für die entstandenen Ergebnisse stellen wir hier nur eine Grafik einer Versuchsperson dar (Abb. 8.5), die den Verlauf der Klassifikationsleistung in Abhängigkeit der Frequenz zeigt.

Trotz der guten Erkennungsleistung jenseits von 50 Hz beschränkten wir uns für die nachfolgenden Versuche auf Frequenzen unter 40 Hz. Einerseits aufgrund des eingesetzten Bandpassfilters, dessen Tiefpass bei 35 Hz beginnt, andererseits aufgrund der

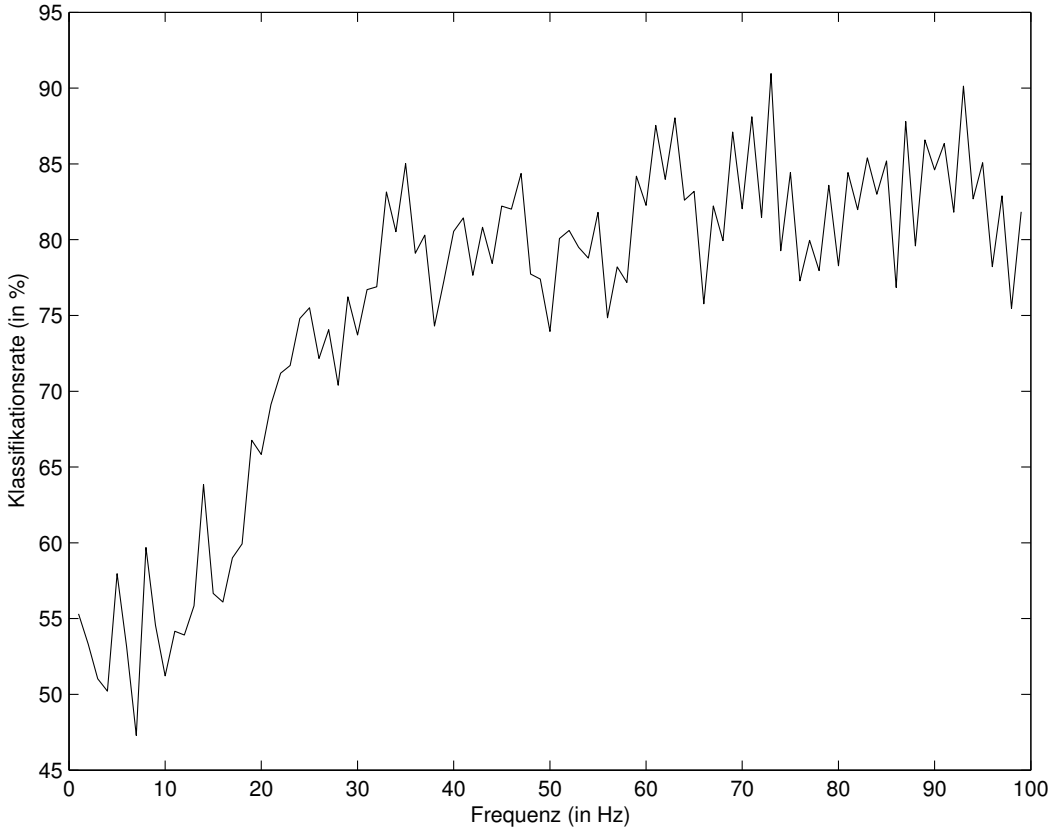

Abbildung 8.5.: Klassifikationsleistung von *OMMbest* für Vp 3 und alle Elektroden von 1-100Hz. Deutlich zu erkennen ist der Anstieg ab ca. 15 Hz, welcher erst bei ca. 35 Hz aufhört. Das Maximum der Klassifikationsleistung liegt sogar noch viel höher (bei ca. 72 Hz).

umstrittenen Bedeutung derartig hoher Frequenzanteile in EEG-Signalen.

8.1.3. Untersuchung nach Elektroden

Durch Auswahl nur einer der 19 vorhandenen Elektroden sind wir in der Lage, positionsspezifische Informationen der Hirnaktivitäten zu erkennen und dokumentierte Resultate zu verifizieren bzw. interessante neue Phänomene zu entdecken. Zu diesem Zweck untersuchten wir für jede Versuchsperson jede einzelne Elektrode im vielversprechendsten Frequenzband von **30-40 Hz** sowie gängige Kombinationen zur Abbildung von ganzen Hirnarealen.

Aus Platzgründen stellen wir nur ein detailliertes Ergebnis einer Versuchsperson dar (Vp 3, s. Tab. 8.5) und begnügen uns ansonsten mit der Zusammenfassung (s. Tab. 8.4). Die besten vier Einzelelektroden kombinierten wir zur weiteren Steigerung der Erkennungsleistung bei einfach zu realisierenden praktischen Anwendungen.

Nach der Mittelung über alle Versuchspersonen erhalten wir für das Frequenzband **30-40 Hz** die in Abbildung 8.6 dargestellte Klassifikationsleistung jeder einzelnen Elektrode.

Elektrodenkombination	Elektroden	Klassifikationsleistung
Front	Fp1, Fp2	84.85 %
beste kombiniert	Fp1, Fp2, F7, T3	**90.12 %**
links anterior	Fp1, F7, T3	84.37 %
rechts anterior	Fp2, F8, T4	87.06 %
links posterior	T5, O1	82.34 %
rechts posterior	T6, O2	80.33 %
hinten	O1, O2	77.78 %

Tabelle 8.4.: Klassifikationsleistung für *OMMbest* auf verschiedenen Elektrodenkombinationen gemittelt über alle Versuchspersonen. Artefaktbehaftete Trials wurden in diesem Fall nicht entfernt.

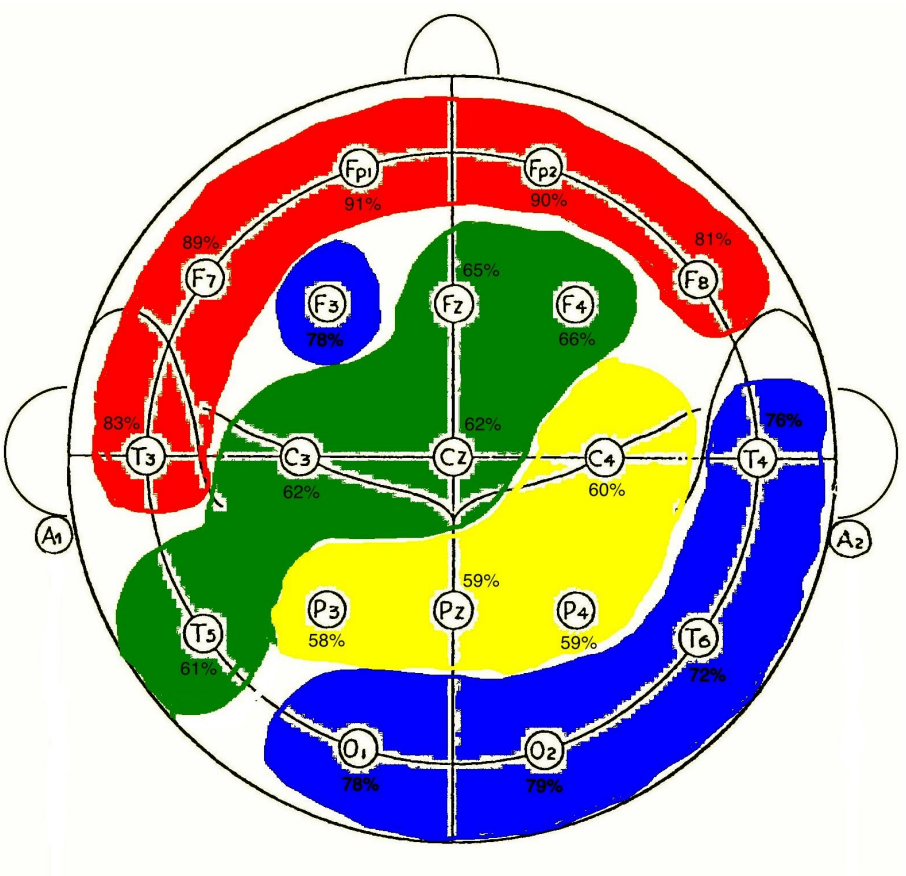

Abbildung 8.6.: Klassifikationsleistung für einzelne Elektroden im Bereich **30-40 Hz**. Bereiche ähnlicher gemittelter Erkennungsleistung sind gleich eingefärbt. Die Intervalle reichen von *über 80 %* (rot) über *70 - 80 %* (blau) und *60 - 70 %* (grün) bis *unter 60 %* (gelb). Deutlich zu erkennen ist das sehr gute Abschneiden der linken vorderen (anterior) und das immer noch gute Ergebnis der hinteren (posterior) Elektroden.

Elektrode	artefaktbereinigt	nicht artefaktber.
1 (FP1)	95.78%	**100.00%**
2 (FP2)	**98.00%**	**98.00%**
3 (F7)	91.56%	**97.78%**
4 (F3)	81.33%	**87.33%**
5 (FZ)	64.44%	**65.11%**
6 (F4)	58.44%	**69.33%**
7 (F8)	**85.33%**	78.67%
8 (T3)	94.00%	**98.00%**
9 (C3)	52.44%	**64.67%**
10 (CZ)	**54.00%**	52.00%
11 (C4)	**58.44%**	56.00%
12 (T4)	**83.56%**	79.33%
13 (T5)	**64.67%**	62.44%
14 (P3)	58.44%	**60.44%**
15 (PZ)	**54.22%**	**54.22%**
16 (P4)	**56.44%**	41.78%
17 (T6)	**69.11%**	68.67%
18 (O1)	**83.11%**	79.33%
19 (O2)	68.44%	**83.56%**
Avg.	72.20%	**73.51%**

Tabelle 8.5.: Klassifikationsergebnisse für *OMMbest* auf EEG pro Elektrode (beste Ergebnisse aus CV für Vp 3). Die Entfernung der artefaktbehafteten Trials hat eher eine negative Auswirkung auf die Erkennungsleistung.

8.1.4. Generalisierungsfähigkeit

Zur Überprüfung der Generalisierungsfähigkeit – d.h. der Möglichkeit, dieselben Muster auf unbekannten Versuchspersonen zu erkennen – trainierten wir ein Modell auf insgesamt 12 der 17 Versuchspersonen im Frequenzbereich von **30-40 Hz**. Die restlichen 5 Vps wurden zum Testen zurückgehalten. Die folgenden Tabellen geben die Kreuzvalidierungsleistungen für die 12 Trainingssubjekte (Tab. 8.6 und 8.8), bzw. die Testleistung auf den 5 Testsubjekten (Tab. 8.7 und 8.9) an. Hierbei kann zudem die Auswirkung der Hyperparameter K und λ von *OMMbest* auf die Erkennungsleistung beobachtet werden.

Zu sehen ist, dass die OMMs besser mit den neuen Beipieldaten zurechtkommen und hyperparameterstabiler sind. Die SVM-Parameter für die optimale Generalisierungsperformanz ($\gamma = 0.00005, C = 0.1$ bzw. $C = 0.05$) liegen in einem anderen Bereich als die besten Kreuzvalidierungsparametern ($\gamma = 0.001, C = 1$), bei welchen keine Generalisierung für neue Versuchspersonen vorliegt. Zwar holt die SVM bei der Generalisierung auf neuen Versuchspersonen auf, jedoch ist für realistische Anwendungsszenarien (z.B. *Brain-Computer-Interface*, BCI) ein Training auf den letzlich zum Einsatz erwählten Personen ratsam. Einerseits sind EEG-Signale sehr personenspezifisch, andererseits ist

K/λ	0	0.1	0.2	0.3
10	68.49%	**69.09%**	67.62%	68.14%
20	70.48%	70.83%	72.05%	**72.66%**
30	74.31%	74.56%	73.52%	**75.87%**
40	**76.74%**	75.09%	74.48%	75.00%
50	74.48%	**75.09%**	73.96%	74.05%
75	73.26%	71.87%	72.83%	**73.53%**

Tabelle 8.6.: Klassifikationsergebnisse der Kreuzvalidierung für *OMMbest* auf EEG-Spektrogrammen für 12 Vps. Artefaktbehaftete Trials wurden nicht entfernt. Man erkennt, dass durch Einsatz des Regularisierungsparameter kleinere Modelle (K=30) ähnlich gute Raten wie größere (K=40) ohne Regularisierung erzielen können. Kleinere regularisierte Modelle versprechen eine bessere Generalisierung. Die fett gedruckten Ergebnisse markieren die Reihenmaxima.

K/λ	0	0.05	0.1	0.2	0.3
20	55.00%	56.67%	54.37%	**62.29%**	57.92%
30	55.00%	**55.62%**	48.54%	53.33%	51.46%
40	49.79%	50.83%	51.67%	**55.21%**	53.75%
50	51.88%	53.33%	54.79%	**59.79%**	57.92%

Tabelle 8.7.: Klassifikationsergebnisse der Generalisierung für *OMMbest* auf EEG-Spektrogrammen für 5 Testpersonen. Für die Modellbildung wurden die Daten der 12 Trainingssubjekte mit Hyperparametern in einem Bereich um das Optimum aus Tabelle 8.6 verwendet.

γ/C	0.5	1	5	10	50	100
1e-05	48.87%	48.87%	59.99%	60.51%	**62.24%**	61.28%
5e-05	51.13%	59.47%	60.85%	**63.02%**	60.24%	60.24%
1e-04	57.29%	60.07%	**62.94%**	62.85%	59.72%	60.16%
5e-04	60.68%	63.03%	**64.06%**	61.98%	62.50%	62.50%
1e-03	60.76%	**64.33%**	63.63%	63.72%	63.72%	63.72%
5e-03	48.87%	56.16%	**59.64%**	**59.64%**	**59.64%**	**59.64%**

Tabelle 8.8.: Kreuzvalidierungsergebnisse der SVM für 12 Versuchspersonen.

γ/C	0.05	0.1	0.5	1	5
1e-05	**58.33%**	**58.33%**	**58.33%**	**58.33%**	56.67%
5e-05	**59.17%**	**59.17%**	58.54%	56.46%	54.37%
1e-04	**57.29%**	**57.29%**	56.67%	55.83%	52.08%
5e-04	**55.42%**	**55.42%**	54.37%	51.67%	49.58%
1e-03	53.96%	53.96%	**54.58%**	52.29%	48.96%

Tabelle 8.9.: Generalisierungsleistung der SVM für 5 Testpersonen.

die Datenakquise in den meisten Fällen nicht umständlicher als der Verwendungseinsatz. Die hier nicht detailliert dargestellten Ergebnisse für die Generalisierung auf dem β_1-Band (13-18 Hz) sprechen ebenfalls für die OMMs (57.12 % in der Kreuzvalidierung gegenüber 53.48 % für die SVM).

8.1.5. Interpretation der Zuordnungen

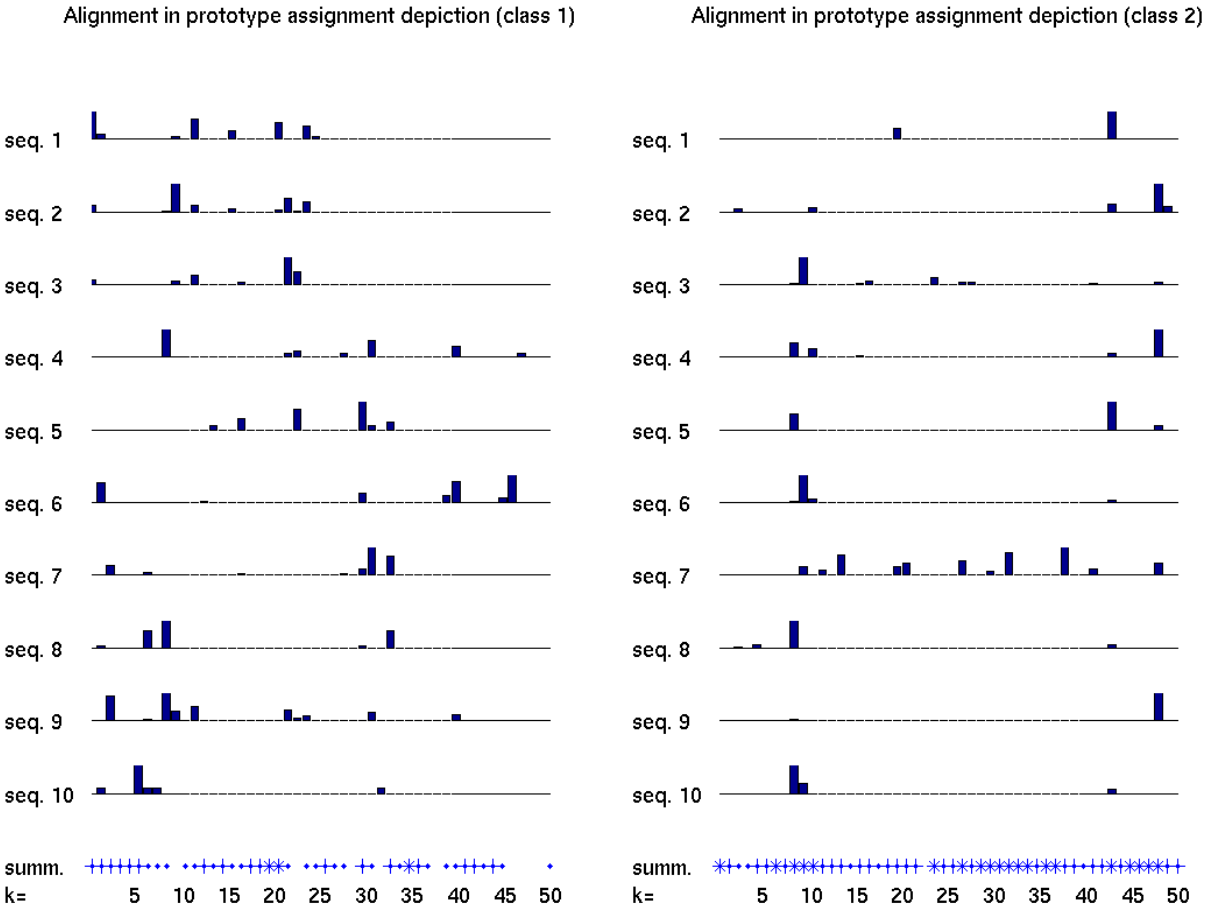

Abbildung 8.7.: Veranschaulichung der aus dem Lernprozess vom *OMMbest* hervorgegangenen Zuordnungen. Je Sequenz (aus Platzgründen nur jeweils 10 pro Klasse) und Prototypvektor (in horizontaler Richtung aufsteigend angeordnet) wird die Anzahl der gelernten Zuordnungen durch die Balkenhöhe dargestellt. In der Konsensuszeile („summ.“) kann die Konserviertheit (s. Text) der Prototypen abgelesen werden. Besonders konservierte Positionen sind durch ein Sternsymbol (*), schwach konservierte Positionen durch einen Punkt, mittlere durch ein Pluszeichen gekennzeichnet. Links sind die ersten 10 Sequenzen der Klasse *abstrakte Wörter*, rechts diejenigen der Klasse *konkrete Wörter* visualisiert.

Die Interpretation der Alignments erfolgt ähnlich wie bei *ClustalW* (s. Abschnitt 3.3) unter Zuhilfenahme einer Konsensuszeile, in welcher die Konserviertheit der Bereiche durch verschiedene Symbole dargestellt wird. Die Konserviertheit ist ein Maß für die Ähnlichkeit mehrerer Sequenzen untereinander. Werden viele gleichartige Merkmalsvektoren bzw. Symbole einer Position zugeordnet, so ist diese Position stark konserviert. Zudem kann auch die Anzahl der Zuordnungen je Prototypvektor und Sequenz veranschaulicht werden (s. Abb. 8.7).

8.2. Proteindaten

8.2.1. Vergleich der Methoden

Um zunächst die grundsätzliche Eignung der Methoden für Proteine zu überprüfen, führten wir auf allen Trainingsdaten des SCOPSUPER95_66-Datensatzes eine Kreuzvalidierung durch. Als Parameter für *FAMmean* wählten wir

$$D \in \{150, 165, 180, 195, 210, 225, 240, 255, 270, 285, 300, 315, 330\}.$$

Für die *FASVM* wählten wir weniger D aus, da zusätzlich noch die ν abgesucht werden mussten. Der Suchraum in diesem Fall war $D \in \{180, 195, 210, 225, 240, 255, 270, 285, 300\}$ und $\nu \in \{0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99\}$.

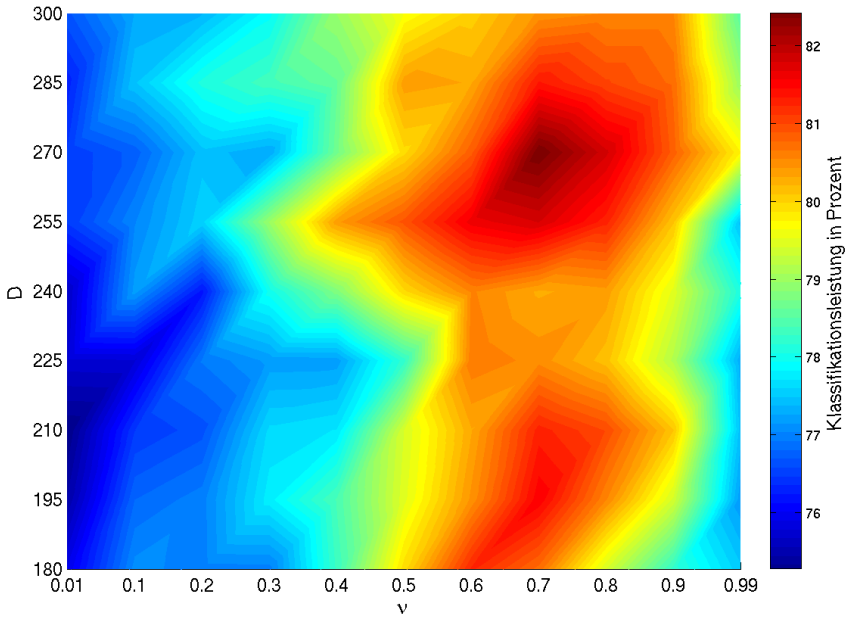

Abbildung 8.8.: Dargestellt ist die Ergebnislandschaft der 5-Fold Kreuzvalidierung auf dem SCOPSUPER95_66-Datensatz. Zum Training wurde die Ein-Klassen-Variante der *FASVM* herangezogen.

In Abbildung 8.8 ist beispielhaft der Hyperparameterraum für den Fall einer *FASVM* mit Indexvektoren aufgezeichnet. Ähnliche Abbildungen existieren auch für die übrigen Methoden, allerdings beschränken wir uns hier aus Platzgründen auf die Ergebnisse der besten Hyperparameter in Tabelle 8.10.

Methode/Merkmale	Blosum	Index
FAMmean	70.98 % ($D = 330$)	77.86 % ($D = 285$)
FASVM	76.07 % ($D = 270, \nu = 0.6$)	**82.47 %** ($D = 270, \nu = 0.7$)
OMMall	65.48 % ($K = 150, \sigma = 1.05$)	71.96 % ($K = 300, \sigma = 1.2$)
OMMbest	65.02 % ($K = 200, \lambda = 0$)	64.38 % ($K = 300, \lambda = 0.05$)

Tabelle 8.10.: Dargestellt werden die Ergebnisse der 5-Fold Kreuzvalidierung der verschiedenen Methoden und Merkmalsvektoren des SCOPSUPER95_66-Datensatzes.

Die korrespondierende Generalisierungsperformanz auf den unbekannten Testdaten ist in Tabelle 8.11 aufgetragen. Zusätzlich angefügt sind die Ergebnisse der besten veröffentlichten Methode (*SCFB BLR HMMS (MLLR)*, s. [Plö05]) und der Standardmethode zum maschinellen Lernen von Proteinen (*PHMM*, s. [Plö05]).

Methode/Merkmale	Blosum	Index
FAMmean	72.97 %	75.44 %
FASVM	76.07 %	**81.45 %**
OMMall	63.96 %	71.96 %
OMMbest	51.59 %	59.01 %
PHMM	**67.1 %**	
SCFB BLR HMMS (MLLR)	**83.2 %**	

Tabelle 8.11.: Die Generalisierungsperformanz auf den unbekannten Testdaten des SCOPSUPER95_66-Datensatzes in Abhängigkeit von Methode und Merkmalsvektoren. Zusätzlich abgebildet sind die Ergebnisse einer merkmalsoptimierten Methode (*SCFB BLR HMMS (MLLR)*) und der Standardmethode (*PHMM*).

8.2.2. Visualisierung eines Prototypen W

Die repräsentative Eigenschaft der FAM-Prototypen legt es nahe, diese nicht nur als Klassifikationsgrundlage zu verwenden, sondern sie einer gesonderten Analyse zu unterziehen. Insbesondere die Indexvektoren geben Einblicke in eine repräsentierte Proteinfamilie, die weitere Erkenntnisse über die Sequenzen versprechen. Abbildung 8.9 zeigt beispielhaft einen solchen Prototypen. Es ist eine deutliche Höhergewichtung einzelner Komponenten in den Prototypvektoren zu erkennen.

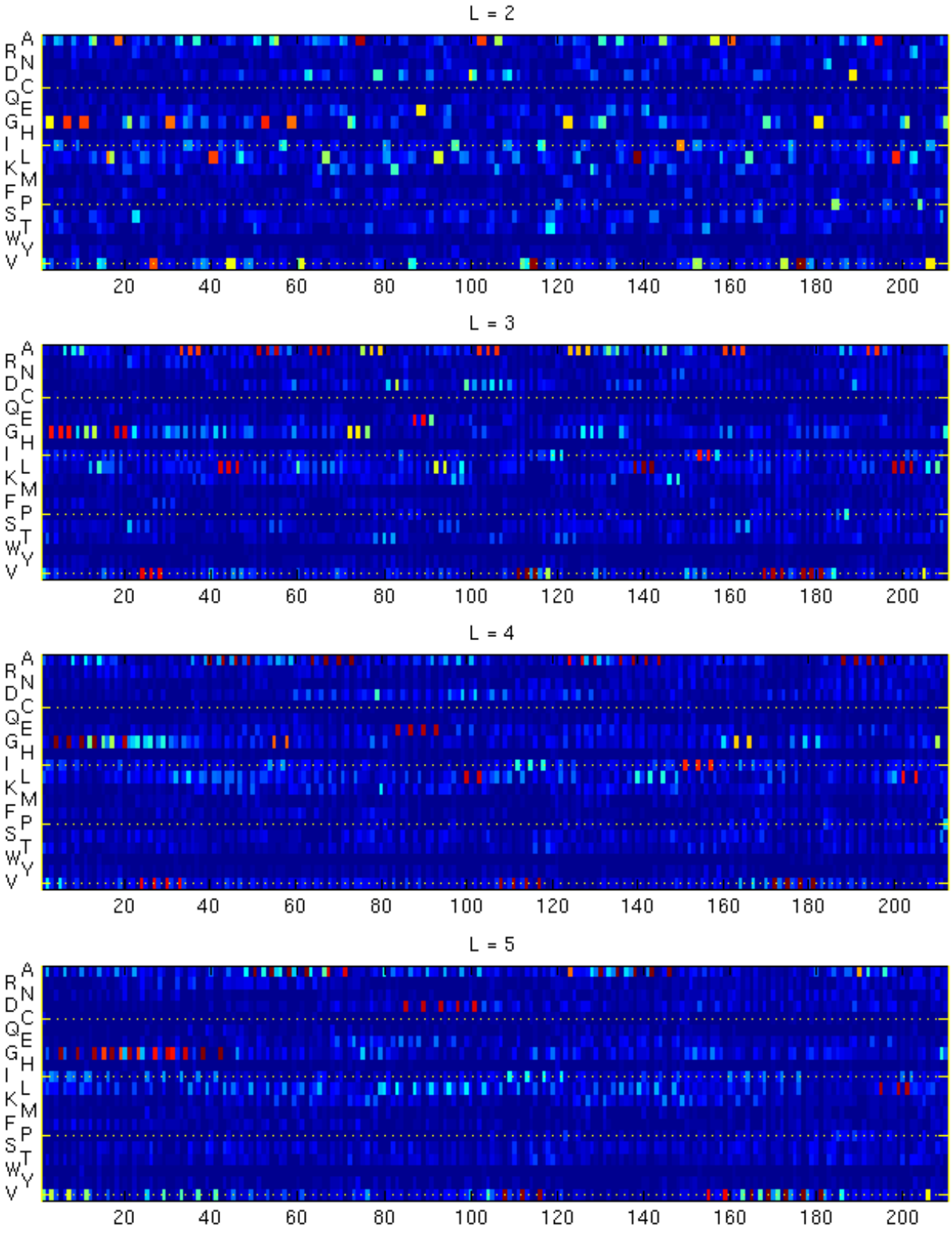

Abbildung 8.9.: Visualisierung der Prototypmatrix $\mathbf{W}$ einer *FASVM*, Klasse 5 des SCOPSUPER95_66-Datensatzes mit Indexvektoren unter den verschiedenen L, trainiert mit $D = 270$ und $\nu = 0.7$.

Der so repränsentierte Prototyp kann in ein Protein zurücktransformiert werden, indem jede Prototypenposition entsprechend ihrer maximalen Komponente in eine Aminosäure übersetzt wird. In Abbildung 8.10 ist dieses Vorgehen beispielhaft für den in Abbildung 8.9 dargestellten Prototypen aufgezeichnet. Die sechs höchsten Werte des Prototypen wurden von uns nochmal fett gedruckt, die entsprechenden Prototypenmuster durch Großbuchstaben hervorgehoben.

L = 2: av vi iy yg gr vn nn tl la ag gt il lf fg gg gi id dt tl lp pd dl lg gi fs sv kg gy ya ad dn ng gk kq tl lq la ag gi ik kv vi id dv fn nh sa an ey yn ng gk gl ld W**D** **D**W vn ne dg gs sl ln nl ln nd de lv ya af G**A** **A**A aa ad dp pd dv va an ns sl ld dy yn vd df fy tg ge ga dn na an ps sv is sv ya ag gn ng ge el ld da ia ag gl lt te nd dg ga ad dl lv vl ll lt tp lg G**G** **G**T wg gd ad dn ng ga vs sd gt el lr ra av

L = 3: aav ang ngr grn nfl ldg dgk gkg fgg gge gel els lla slg lgg ggn gnn qng ngn gnd lik iky kvv vla lag avi ail iid adf dfv dsa say ayv ydd ndl dln lne nea vdg dgl gyl lld **L**DN lva gaf GY**A** Y**A**A **A**AP aid ida daa akn kns nsd sdy dyv idg dgv gip vln gns nsp spd svn vfa faa aal ali laa aas asg egg ggt gtp ada dav qsl sla lav VL**G** L**G**G ggd gdn dng ngd gdt dtn anl nli lir

L = 4: avdg dgrn gnnl ngdp afag fagl agld gldl ylsd lsdl sslg slgg lggn ggng qngn ngnd gvdv vgvd nvla vdia diad aads adsn vanh anel nflg flgd eldg ldgg dggd ggdl nlda ldna araa raad aadp adpn daga akns vasa asld sldg vdgv dgvg gvrf gyns yssg ssgn sgan gank aael aala lata apaa pagd agdn lgad Q**A**DL **A**SLD nllg GEG**G** EG**G**V G**G**VN **G**VNF sgdt tael nels lsnd

L = 5: sfyng fkngn kngtn agadg aadgl adglk dglsd llsdg lslgg slggn lggng ggngf gkgvr kgvra gvraa niaav iaavv aavrs avrsl ahadg hadgl fdggd dggdl ggdld rllde lldaa ldaaa daadp aadpa adpaa dakld ayldy yldyi ldgig dgigs ndlsg dlsgg lsggl **S**GGLN gvnvs anlgl nnsaa SA**A**GA A**A**GAL **A**GALA aalas alayg ngegg gvggv W**G**GVD **G**GVDP dfgna asnel liptr

Abbildung 8.10.: Rückübersetzung des Prototypen aus Abbildung 8.9 in Aminosäuresequenzen in Abbhängigkeit der verschiedenen Musterlängen L. Die durch Fettdruck hervorgehobenen Aminosäuren entsprechen den maximalen Komponenten aus Abbildung 8.9, die Großbuchstaben heben nochmal die zugehörigen Prototypenmuster hervor.

8.2.3. FAM-Hauptkomponentenanalyse

Eine Hauptkomponentenanalyse der Sequenzen des SCOPSUPER95_66-Datensatzes ist in Abbildung 8.11 zu sehen. Beispielhaft zeigen wir die Trainingssequenzen der 7. Klasse gegen die zu einer Menge zusammengefassten Trainingssequenzen der übrigen Klassen. Alle Sequenzen wurden gegen den Prototypen der 7. Klasse aligniert, welcher mit $\nu = 0.7$ und $D = 270$ einer *FASVM* trainiert wurde.

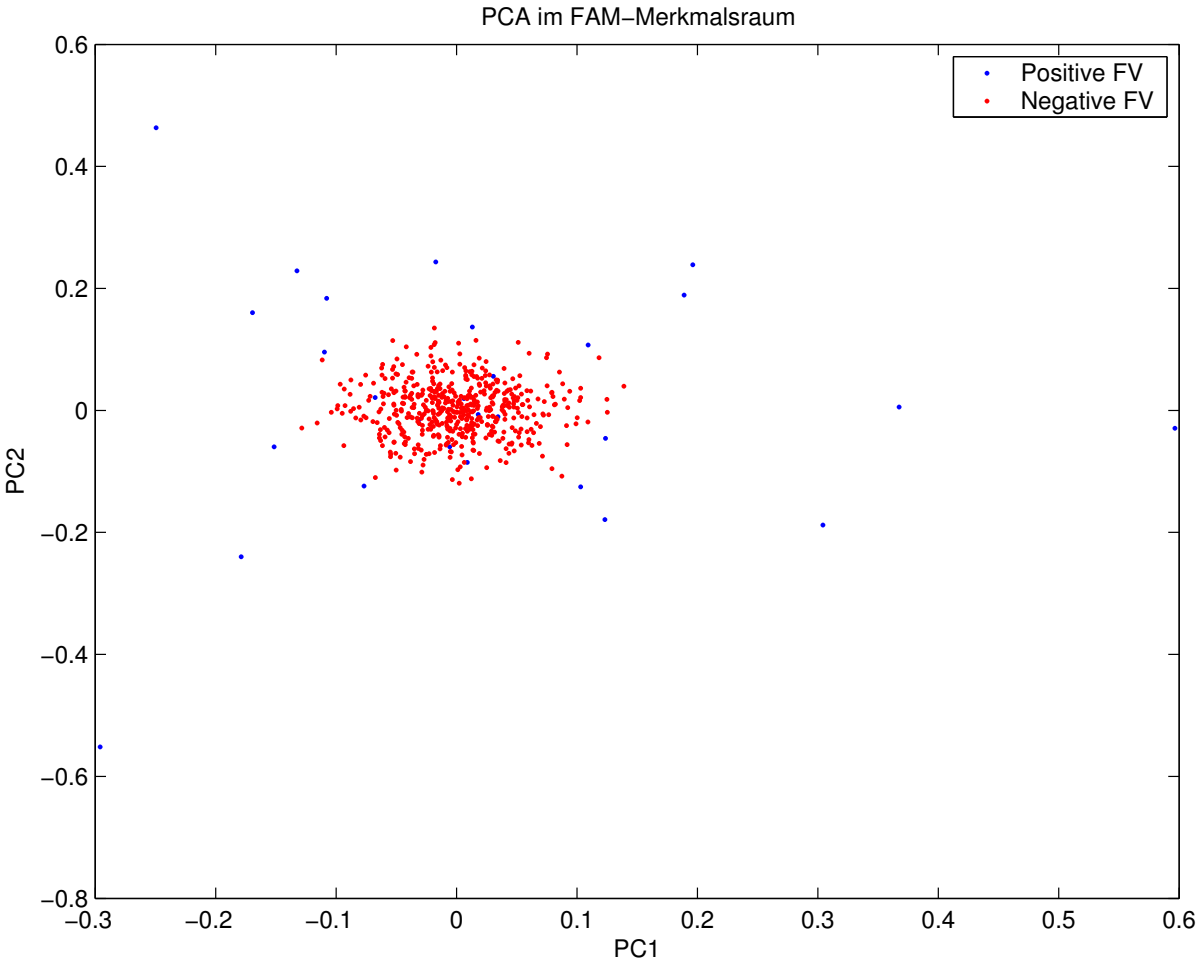

Abbildung 8.11.: Visualisierung der ersten beiden Hauptkomponenten der SCOPSUPER95_66-Daten. Positive Trainingsdaten sind alle positiven Trainingssequenzen der 7. Klasse, negative Trainingsdaten sind die zu einer Menge zusammengefassten Trainingsbeispiele der anderen Klassen. Zu erkennen ist, dass die positiven Testbeispiele eine große Streuung aufweisen, während die negativen Testbeispiele aufgrund der kleineren Scores relativ eng beieinander liegen. Als $\mathbf{w}$ wurde der Prototyp der positiven Trainingsdaten gewählt, welcher mit $D = 270$ und $\nu = 0.7$ trainiert wurde.

8.2.4. Visualisierung der Zuordnungen

Die im vorigen Abschnitt für EEG-Daten vorgenommene Visualisierung der gelernten Zuordnungen bzw. Verantwortlichkeiten lässt sich analog auf die SCOPSUPER95_66-Daten anwenden (s. Abb. 8.12). Hier kann die Konserviertheit der Positionen sogar benutzt werden, um strukturelle und funktionale Eigenschaften der Proteine abzuleiten bzw. um Motive ausfindig zu machen.

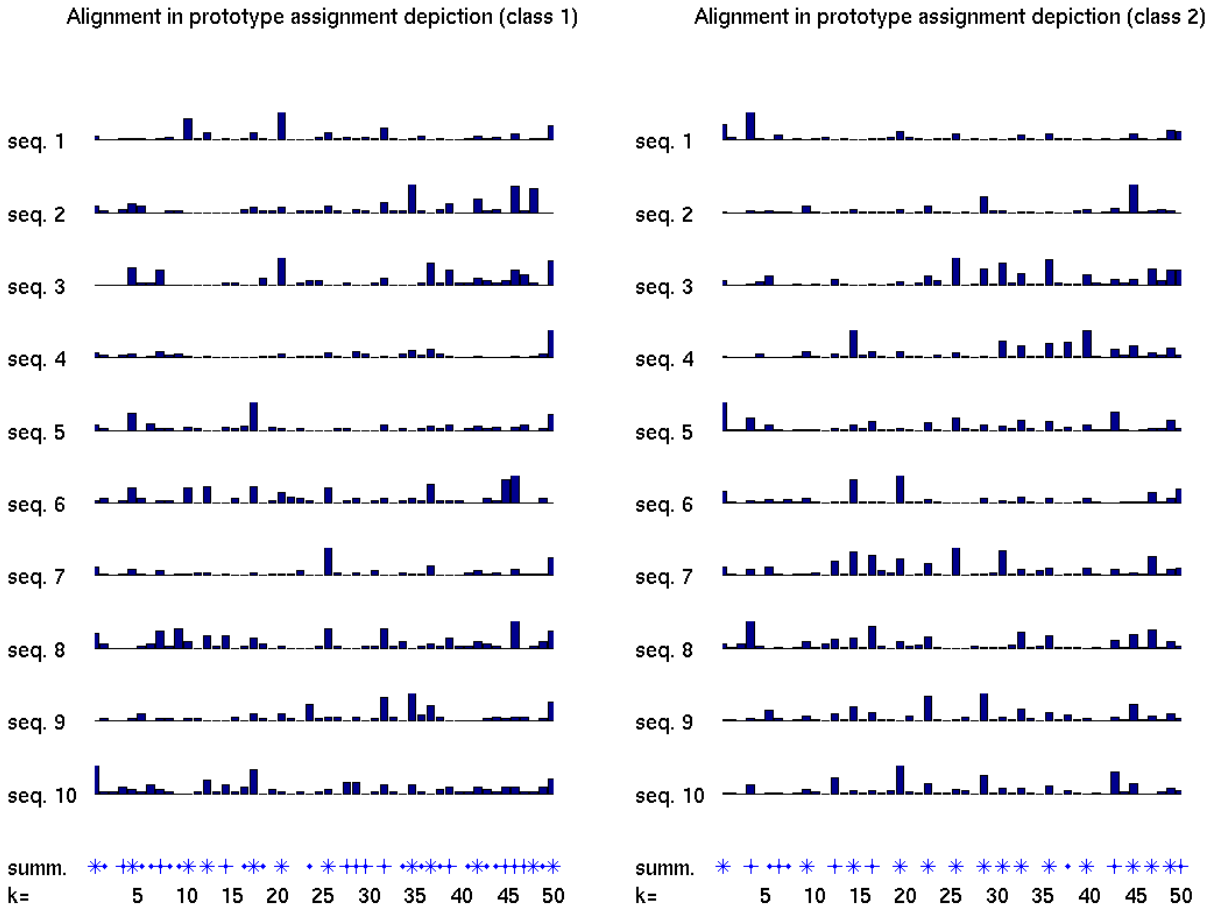

Abbildung 8.12.: Veranschaulichung der aus dem Lernprozess vom *OMMbest* hervorgegangenen Zuordnungen (wie in Abb. 8.7). Links sind die ersten 10 Sequenzen der 1. Klasse, rechts diejenigen der 2. Klasse des in Abschnitt 7.1 beschriebenen Datensatzes visualisiert.

9. Diskussion und Interpretation

9.1. Ordered-Means-Modelle

Mit den *Ordered-Means-Modellen* wurde ein schneller und einfach zu verwendender Ansatz präsentiert, um Positionszuordnungen und generative Modelle aus verschieden langen Sequenzen zu lernen. OMMs stellen dabei eine konsequente Weiterentwicklung der HMMs dar, die mit ihren explizit zu spezifizierenden und zu lernenden Übergangswahrscheinlichkeiten unter Umständen mehr Parameter als notwendig in das Modell einbringen und damit evtl. sogar einen erfolgreichen Lernprozess verhindern.

Die zeitliche Korrelation der Sequenzdaten äußert sich bei den OMMs durch die Entstehung von *Glättungsreferenzvektoren*, welche im Alignmentschritt viele Sequenzvektoren auf sich ziehen und damit zu Sammelstellen uninteressanter Daten werden. So können die wichtigen Informationen in wenigen aussagekräftigen Prototypvektoren modelliert werden.

Die Evaluation zeigt, dass die OMMs in der Lage sind, sowohl interpretierbare als auch leistungsfähige Modelle zur Klassifikation von EEG-Signalen ereigniskorrelierter Potentiale zu lernen. Wichtig dabei ist, dass diese Modelle im Gegensatz zur weit verbreiteten Standardmethode, der Mittelung, auch *online* einsetzbar sind, d.h. direkt nach der Reizpräsentation ausgewertet werden können.

Durch die Zielfunktion (s. Kap. 5) ist mit genügend hoher Rechengenauigkeit die Konvergenz beider Varianten gewährleistet. Dabei kann *OMMall* für kritische σ – d.h. für solche Glättungsparameter, für die die Berechnung der Produktionswahrscheinlichkeit numerisch instabil wird – durch *OMMbest* ersetzt werden. Durch die eingesetzte Initialisierung auf Basis der gemittelten, geglätteten Trainingssequenzen werden lokale Optima (wie beim Einsatz von multiplen Alignments in PHMMs) vermieden.

Wie an den Ergebnissen aus den Tabellen 8.6 bis 8.9 zu sehen ist, generalisiert die deterministische Variante auf den verwendeten EEG-Daten besser als die Support-Vektor-Maschinen. Dies ist vermutlich auf die Berücksichtigung der Positionsvariabilität und Korrelationen in den Sequenzen zurückzuführen.

Der Alignmentschritt der OMMs liefert intuitiv interpretierbare Zuordnungsmuster. So ist z.B. in den Abbildungen 8.1 und 8.2 zu erkennen, dass beim Alignment des abstrakten Nomens die ersten Referenzvektoren viele Sequenzpositionen aufsammeln, während spätere Referenzvektoren weniger Sequenzvektoren auf sich vereinen und somit die zeitliche Entwicklung feiner auflösen. Beim konkreten Nomen verhält es sich genau umgekehrt, was die kognitionswissenschaftliche Vermutung stützt, dass natürlichsprachlich präsentierte, konkrete Wörter ein frühes bildhaftes, abstrakte Wörter hingegen ein verzögertes semantisches Verständnis bewirken (vgl. [Mül03]).

Die mit OMMs produzierbaren multiplen Alignments – welche in Abbildung 8.7 beispielhaft dargestellt sind – liefern Aussagen über wichtige und weniger wichtige Bereiche durch die Anzahl der gelernten Zuordnungen auf einen Prototyp bzw. die Verteilung der Verantwortlichkeiten.

Die Modelle bestehen aus einer Matrix mit Mittelwertvektoren (Prototypen, Prototypvektoren) in den Spalten. Durch Inspektion dieser Prototypen (Abb. 9.1) können Aktivitäten in bestimmen Frequenzbändern festgestellt werden, was bei der Mittelung nicht möglich ist (s. [WM03]). Dabei generalisieren kleinere Modelle besser als größere (vgl. Tab. 8.7), was unter Berufung auf Lerntheorien (s. Kap. 2) absolut sinnvoll ist.

Die Ergebnisse aus [WR96] – nämlich, dass in für kognitive Prozesse verantwortlich gehaltenen Frequenzbändern abstrakte und konkrete Wörter unterschiedliche Hirnareale zu unterschiedlichen Zeitpunkten stimulieren – sind nachvollziehbar, auch wenn dort nur das α_1- und das β_1-Band untersucht wurden. Daher vermuten wir einen direkten Bezug von OMMs zur Kohärenz (s. [WM03]), welche ein statistisches Maß für die Korrelation zweier EEG-Signale in einem bestimmten Frequenzband ist. Allerdings ist die Kohärenz aufwändig zu berechnen und sehr parametersensibel.

Die guten Leistungen in den hohen Frequenzen (s. Abb. 8.5) sind entweder auf Muskelaktivitäten (z.B. Stirnrunzeln) oder kognitive Prozesse im oberen γ-Band zurückzuführen. Eine mögliche Erklärung für die schlechtere Leistung in den unteren Frequenzbändern könnte die Maskierung wichtiger Informationen durch primärsensorische – also nichtkognitive – Prozesse höherer Intensität sein. Systematische Fehler sind nahezu auszuschließen, da die Versuche streng randomisiert (sowohl die Teilversuche abstrakt/konkret als auch die Reihenfolge der Wörter) stattfanden und die Signale von hoher elektrischer Güte sind. Vermutlich spielen sich in diesen hohen Frequenzen mehr kognitive Prozesse ab als bisher angenommen. Zukünftige Experimente mit dem Ziel, dies herauszufinden werden hoffentlich Aufschluss darüber geben.

Unsere Vermutungen darüber, warum das Verfahren so gut auf den untersuchten EEG-Signalen funtioniert, gehen dahin, dass die Positionsvariationen in den deterministischen Zuordnungen repräsentiert werden und die weniger wichtigen Bereiche in einigen Glättungsreferenzvektoren gesammelt werden.

Die Klassifikationsergebnisse für die Proteindaten waren insgesamt nicht so hervorragend, dennoch sind die OMMs hier mit den PHMMs vergleichbar bzw. leicht besser. Zudem werden keine multiple Alignments als Initialisierung benötigt, was einen teuren und langwierigen, vorhergehenden Schritt spart. Im Vergleich der Methoden (s. Tab. 8.11 und [PF04]) erreicht *OMMbest* mit 59.01 % nicht das Niveau der PHMMs, *OMMall* übertrifft diese mit 71.96 % Klassifikationsrate jedoch deutlich, reicht aber nicht an die Ergebnisse der besten veröffentlichen Methoden heran. Aufgrund der relativ groben Abtastung der Hyperparameter und dem Einsatz naiver Merkmalsvektoren sind allerdings noch weitere Verbesserungsmöglichkeiten zu erwarten.

Die Ergebnisse unterstreichen die domänen- und anwendungsübergreifende Leistungsfähigkeit dieses Ansatzes und stellen die Verwendung von Übergangswahrscheinlichkeiten und initialen multiplen Alignments und den damit verbundenen Annahmen über die Daten in Frage.

Die Interpretation der entstandenen multiplen Alignments (z.B. Abb. 8.12) gibt idea-

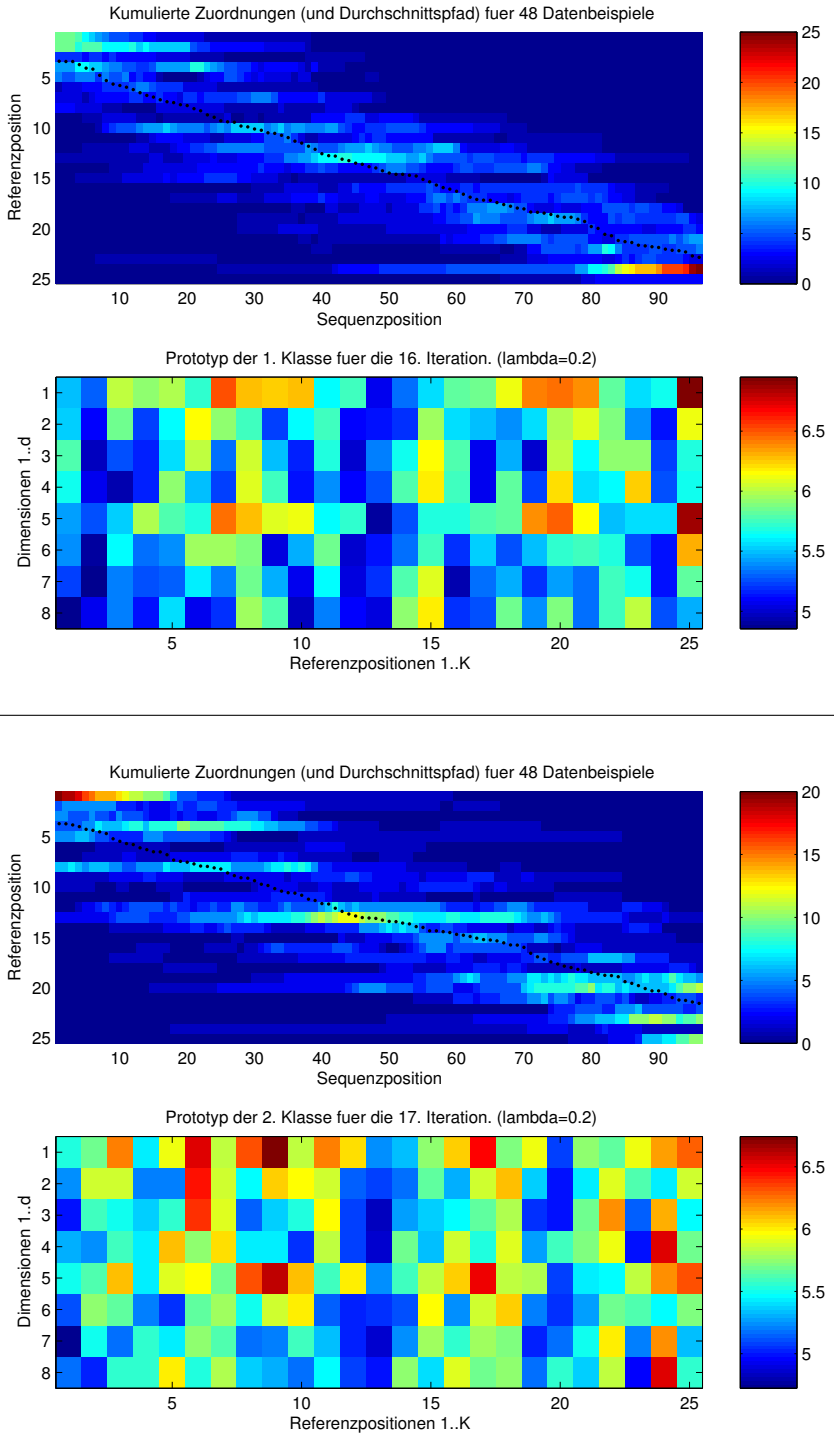

Abbildung 9.1.: Veranschaulichung der Zuordnungen und Prototypen von *OMMbest* für ein abstraktes Wort (obere 2) und ein konkretes Wort (untere 2) für das β_1-Band (13-18 Hz) und die beiden Frontelektroden (Fp1, Fp2) nach dem jeweils letzten Iterationsschritt. Man erkennt die unterschiedlichen Gewichtungen für die einzelnen Frequenzkoeffizienten (Dimension 1 = 12-13 Hz von Fp1, Dimension 8 = 18-19 Hz von Fp2) sowie Referenzvektoren mit kleinen Gewichten (fast homogen blaue Spalten). Die Paramter betragen in diesem Fall $K = 25$, $\lambda = 0.2$.

lerweise Aufschluss über konservierte Bereiche einer Proteinfamilie. Da die Visualisierung nicht Hauptbestandteil der Arbeit war, gehen wir davon aus, dass durch intensive Beschäftigung unter Einbeziehung von Experten in naher Zukunft wesentlich intuitiver interpretierbare multiple Alignments und Identifikationen konservierter Bereiche auf der Grundlage von OMMs zu erwarten sind.

9.2. Feature-Alignment-Maschinen

Feature-Alignment-Maschinen sind leistungsfähige Werkzeuge zur Sequenzanalyse, die sich – wie die OMMs – durch einfache Handhabung auszeichnen.

Die Ergebnisse der Experimente zeigen, das FAMs auf den diskreten Proteindaten sehr gut klassifizieren und auch die Klassifikationsperformanz auf den EEG-Daten ist überdurchschnittlich.

Die Generalisierungsperformanz der *FASVMs* auf den unbekannten Testdaten des SCOPSUPER95_66-Datensatzes entspricht den Erwartungen, die an ein modernes Klassifikationssystem gestellt werden können. Mit einer Klassifikationsrate von 81.45 % (für *FASVM* mit Indexvektoren, vgl. Tabelle 8.11) zählen FAM-Klassifikatoren zu den fortschrittlichsten Verfahren in diesem Bereich. Die Ergebnisse der *SCFB BLR HMMS (MLLR)* ([Plö05]) wurden durch aufwändige Merkmalsextraktion erreicht und sind daher nur bedingt mit den Ergebnissen der von uns verwendeten Indexvektoren vergleichbar. Es bleibt abzuwarten, inwieweit hochentwickelte Merkmale die Ergebnisse von FAMs nochmals verbessern können.

Im Detail schneidet die *FASVM* besser ab als der *FAMmean*-Algorithmus, was durch die Möglichkeit der Ein-Klassen-SVM, einen stabileren Mittelwertschätzer zu realisieren und durch die über den Hyperparameter ν einstellbare Regularisierung erklärbar ist. Die diskriminative Variante der *FASVM* konnte aufgrund der durch den erhöhten Rechenaufwand gröberen Evaluation bisher die guten Ergebnisse der Ein-Klassen-*FASVM* nicht übertreffen.

Die Modellgröße scheint nicht hauptsächlich ausschlaggebend für die Klassifikationsperformanz zu sein. Wie Abbildung 8.8 zeigt, können mit $D = 180$ noch sehr gute Klassifikationsergebnisse erzielt werden, auch wenn das Maximum der Kreuzvalidierungsperformanz bei $D = 270$ liegt. Für $\nu = 0.7$ zieht sich ein Korridor guter Klassifikationsraten über die Modellgrössen hinweg. Da durch den Parameter ν auch der Anteil der Supportvektoren reguliert wird, sind rückwirkende Aussagen über die Verteilung der Daten möglich.

Die FAMs beachten Kurzzeitkorrelationen über den Verfahrensparameter L. Sie fassen nicht, wie die OMMs, zeitliche Zusammenhänge in einem Zustand zusammen, sondern greifen einzelne Positionen, unabhängig von ihrer Nachbarschaft, heraus. Kurzzeitkorrelationen werden durch verschiedene Musterlängen realisiert und können vom Benutzer, je nach Domäne, angepasst werden.

Dieses Verhalten entspricht der Natur der Proteindaten und erklärt die überdurchschnittliche Performanz auf diesen Daten. Die EEG-Daten beinhalten ebenfalls Korrelationen, jedoch sind diese vermutlich langreichweitiger, sodass die Berücksichtigung von

Langzeitkorrelationen erforderlich ist. Dies ist mit dem Mittelwert und der SVM nicht möglich, da die einzelnen Dimensionen der FAM-Merkmalsvektoren bestimmten (latenten) Merkmalen entsprechen. Dass die FAMs trotzdem noch so gute Ergebnisse auf den EEG-Daten erreichen, ist ein weiteres Indiz auf die Leistungsfähigkeit dieser Methode. Langzeitkorrelationen können von FAMs durch die adaptive Merkmalsselektion realisiert werden. Die korrelierten Merkmale der Sequenzen werden im Verlauf des Trainings gelernt und bleiben im Merkmalsraum erhalten. Mit entsprechenden ML-Methoden im Regressionsschritt (z.B. Self-Organizing Maps, s. [Koh97]) können langreichweitige Korrelationen dann ebenfalls analysiert werden.

Der Prototyp der FAMs mit Indexvektoren ist als Folge von Aminosäuren interpretierbar. In Abbildung 8.9 ist zu erkennen, dass einzelne Sequenzvektorkomponenten deutlich höhere Gewichte gegenüber den übrigen Komponenten haben. Die Transformation in eine Aminosäurekette (Abb. 8.10) zeigt, dass sich die Prototypen als Repräsentanten von Proteinsuperfamilien interpretieren lassen. Diese Repräsentanten im Detail zu analysieren und zu interpretieren bleibt zukünftigen Arbeiten überlassen.

Die Visualisierung mit Hilfe einer Hauptkomponentenanalyse im FAM-Merkmalsraum zeigt deutlich unterscheidbare Klassen (Abb. 8.11). Die große Streuung der positiven Testbeispiele erklärt sich durch den höheren Score der Sequenzen, was den Ergebnissen der Klassifikationsanalysen entspricht. Die Erkenntnisse, die durch eine PCA auf Sequenzen ermöglicht werden, müssen ebenfalls in weiterführenden Arbeiten herausgestellt werden. Die Hauptkomponentenanalyse ist eine seit langem etablierte Methode und die Möglichkeit, diese nun auf Sequenzen anzuwenden, ist ein Fortschritt in der Sequenzanalyse.

Die These, dass Merkmalsvektoren mit biologischem Wissen die Klassifikation von Proteindaten verbessern, kann in unseren Analysen nicht bestätigt werden. Im Gegenteil sinkt die Klassifikationsperformanz für die Blosumvektoren gegenüber den Indexvektoren (vgl. Abschnitt 7.1.2) sowohl bei OMMs als auch bei FAMs. Allerdings gibt es in neueren Veröffentlichungen Hinweise (vgl. [PF04], [PF05], [Plö05]), dass durch fortschrittliche Merkmalsextraktion bessere Resultate erzielt werden können.

Zusammenfassend gilt, dass die FAMs eine leistungsfähige Sequenzanalyseumgebung darstellen. Die bisherigen Ergebnisse bieten noch viel Raum für weiterführende Interpretationen und Forschungen. In Zunkunft verspricht besonders die Eigenschaft der FAMs, Algorithmen des maschinellen Lernens ohne Veränderungen der Zielfunktion durch ein FAM-Plugin für Sequenzen lernbar zu machen, weitere, interessante Algorithmen und Ergebnisse. Gegenüber Kernmethoden (s. Abschnitt 3.5) zeichnen sich FAM-Algorithmen durch einen geringeren Rechenzeitbedarf (vgl. Anhang A.3) und eine explizite Merkmalsraumrepräsentation aus. Diese ermöglicht weitere Einblicke in die gelernten Sequenzen, welche Kerne nicht aufweisen.

10. Fazit und Ausblick

Unsere Arbeit zeigt, dass die von uns vorgestellten Methoden in der Lage sind, datengetriebene Alignments von Sequenzen unterschiedlicher Domänen zu lernen. Die Ergebnisse der Proteinklassifikationsexperimente sind für die *Feature-Alignment-Maschinen* trotz Verzicht auf initiale multiple Alignments besser als die der Standardmethoden. Auf den EEG-Daten waren vor allem die *Ordered-Means-Modelle* sehr erfolgreich bei der Klassifikation und bestätigten sowohl veröffentlichte Ergebnisse sowie Vermutungen. Dabei sind wir mit der Erstellung und Interpretation von Alignments auf EEG-Daten einen bisher noch nicht verfolgten Weg gegangen und konnten zeigen, dass solche Untersuchungen wertvolle und neue Informationen hervorbringen können.

Unsere Erwartungen wurden auch in Bezug auf die Interpretierbarkeit der Modelle erfüllt, wobei in diesem Bereich die Einbeziehung von Expertenwissen zu noch aussagekräftigeren Visualisierungen führen könnte. Beide Methoden sind – durch die während dieser Arbeit entstandenen Toolboxen – schnell und einfach zu verwenden, sodass die Durchführung weiterer Experimente auf anderen Datendomänen durch engagierte Forscher möglich ist und von uns erhofft wird.

Aber auch auf den von uns untersuchten Domänen sind noch Experimente zur Verbesserung der Erkennungsleistung denkbar. So könnte bei den EEG-Daten eine Multiskalenanalyse mit Wavelets durchgeführt werden, um zeitlich verschieden aufgelöste Bestandteile des Signals zu berücksichtigen. Auch die Verwendung zeitlich höher auflösender Spektrogramme mit höheren Frequenzanteilen ist mit Hinsicht auf die überraschend guten Ergebnisse im oberen Frequenzbereich wünschenswert. Die Auswertung von Spektrogrammen nach Zeitanteil (erste 250 ms, 500ms, 1s,...) könnte ebenfalls wertvolle Aussagen liefern, da nach gängiger Sprachwahrnehmungstheorie ein Wort meist schon 250 ms nach der Präsentation verstanden ist (vgl. [Mül03]). Weiterhin bietet sich in unserem speziellen Fall die Untersuchung von Daten aus ähnlichen Experimenten mit gleichen Versuchspersonen zu einem späteren Zeitpunkt an, um die Generalisierungsfähigkeit weiter zu analysieren.

Auf dem relativ jungen Gebiet der maschinellen Analyse molekularbiologischer Daten kommen außer dem Einsatz besserer (biologisch motivierter) Merkmalsvektoren auch andere Arten von Daten, wie z.B. DNA- bzw. RNA-Sequenzen, in Frage.

Weitere Domänen mit den zu Anfang dieser Arbeit geschilderten Problemen betreffen die Sprach- und Objekterkennung, Kursdaten aus den Wirtschaftswissenschaften, medizinische Datenaufzeichnungen sowie Bewegungstrajektorien von Robotern. Bei Letzteren sind vor allem die Analyse und Generierung interessant, aber auch die Objektklassifikation mittels Trajektorienanalyse ist denkbar. Dabei könnten mit Bewegungssensoren die haptischen Eigenschaften von Objekten zur Unterscheidung von Materialien und Formen genutzt werden.

Auch die Methoden selbst bieten noch viele Möglichkeiten der technischen Weiterentwicklung. Bei den OMMs sind diskriminative Varianten durch Verwendung des Likelihood-Verhältnisses zweier Klassen und die Erzeugung diskriminativer Fisher-Scores (s. [Jaa98]) als Merkmale zur Weiterverwendung mit anderen ML-Methoden denkbar. Zur Verbesserung der numerischen Stabilität von *OMMall* und der daraus resultierenden Möglichkeit, die Modelle für kleinere σ berechnen zu können, empfiehlt sich der Einsatz von Datentypen mit höherer Rechengenauigkeit als 64 bit.

Durch ihren einheitlichen Merkmalsraum lassen sich die *Feature-Alignment-Maschinen* mit verschiedenen Methoden des ML kombinieren. Einsatzmöglichkeiten sind z.B. die Partitionierung von Datenräumen (Clustering), die Visualisierung, wie wir sie durch Verwendung der PCA schon angeschnitten haben sowie die Datenkompression. Für die Berücksichtigung und Analyse von nichtlinearen Langzeitkorrelationen bietet sich die Anwendung des FAM-Prinzips auf Self-Organizing-Maps (SOM, s. [Koh97]) und die unüberwachte Kernregression (s. [Mei05]) an.

A. Implementation

A.1. Entwicklungsumgebung

Als Entwicklungsumgebung wählten wir Matlab 6.5.0.180913a ([Mat05]). Die Betriebsplattform bildeten verschiedene x86-PCs mit SuSe-Linux ab Version 8.0. Zum einen gestattet Matlab eine schnelle und einfache Umsetzung der mathematischen Grundkonzepte, zum anderen ist die Software sehr verbreitet und ermöglicht damit den unkomplizierten Einsatz und die Erweiterung der von uns entwickelten Toolboxen (s. Abschnitt A.4).

Einige rechenintensive Methoden, wie z.B. die Berechnung der Distanzmatrix für das dynamische Programm der Ordered-Means-Modelle (s. Kap. 5) sowie des Alignment-Schritts beider Verfahren, implementierten wir in der Programmiersprache C. Die Einbindung erfolgt über die C-Schnittstelle von Matlab. Als Versionskontrollsystem benutzen wir CVS (*Concurrent Versions System*) in der Version 1.11.6.

A.1.1. Datenstruktur

Die in Kapitel 4 und Abschnitt 7.1 beschriebenen multivariaten Sequenzen $\mathbf{S}$ bilden jeweils eine $d \times T$-Matrix. Da die Länge der Sequenzen variieren kann, ist eine klassenweise Organisation in Matlab-Feldern (*cell-arrays*) der Größe N_{y_i} erforderlich. Auch die Anzahl der Sequenzen pro Klasse ist variabel und legt somit ebenfalls die Anordnung in cell-arrays nahe. Somit ergibt sich beispielsweise für die dritte Sequenz der zweiten Klasse `sequenz_2_3 = data{2}{3}` als Adresse des Datensatzes `data`. In analoger Weise werden die Prototypen und die Zwischenergebnisse (z.B. Zuordnungsmatrizen) der Funktionen abgelegt.

A.1.2. Programmstruktur

Die Programme sind als Matlab-Funktionen (*m-files*) hierarchisch angelegt. Da sich die beiden Verfahren ähnlich sind, können sie zum Teil die gleichen Funktionen verwenden (z.B. Merkmalsextraktion, Initialisierung).

Die verfahrensspezifischen Methoden sind geschachtelt organisiert, d.h. Alignment und Regression werden von der Trainings- bzw. Testfunktion (nur Alignment) aufgerufen, welche wiederum von der Kreuzvalidierungsfunktion gesteuert werden.

Die funktionsspezifischen Optionen haben wir für jedes Verfahren als Matlab-Struktur (*struct*) implementiert, so dass diese gespeichert und wiederverwendet werden können.

A.1.3. SVM

Für den Vergleich der Methoden bzgl. der EEG-Daten (s. Abschnitt 7.1) benutzten wir die bestehende Matlab-Version 2.8 der *libsvm* ([Lin05]). Diese implementiert bereits die Ein-Klassen-SVM, SMO und die Multiklassenklassifikation (s. Abschnitt 2.7).

A.1.4. Numerische Aspekte der Wahrscheinlichkeitsrechnung

Die Berechnung der Wahrscheinlichkeiten der *Ordered-Means-Modelle* finden konsequenterweise im negativen logarithmischen Raum statt und die Emissionsdichten werden somit zu Distanzen. Dabei helfen einfache Logarithmusrechenregeln – wie der Übergang von Produkten zu Summen – und die *Kingsbury-Rayner-Formel* (vgl. [KR71])

$$\log(a+b) = \log(a) + \log(1+\frac{b}{a}) = \log(a) + \log(1+e^{\log(b)-\log(a)}) \tag{A.1}$$

zur schnellen und numerisch genauen Bestimmung der Summe im logarithmischen Raum. Gilt $a << b$ oder $a >> b$ kann einer der Terme vernachlässigt werden, für viele Programmiersprachen (u.a. auch C) existieren zudem spezielle Funktionen zur Berechnung von $\log(1+x)$.

A.2. Pseudocode

In diesem Abschnitt führen wir die Ablaufstrukturen der wichtigsten (nichttrivialen) Algorithmen der Kapitel 5 und 6 auf. Dabei berufen wir uns auf die in den Kapiteln und in Anhang B beschriebene Notation.

```
1.  for c = 1..M
2.    repeat
3.      for i = 1..N_c
4.        x_i^c = align(S_c^i, W_c)
5.      end
6.      X_c = [x_c^1, .., x_c^N]
7.      W_c = regression(X_c)
8.    until convergence
9.  end
```

Abbildung A.1.: Pseudocode für das Projektions-Regressions-Schema der FAMs.

1. Berechne Skalarproduktmatrix $\mathbf{S}$
2. for $k = 1..K$
3. for $t = 1..T$
4. $\mathbf{s}_{kt} = \max(\mathbf{S}_{kt} + \mathbf{S}_{k-1,t-1}, \mathbf{S}_{k,t-1})$
5. $\mathbf{\Phi}_{kt} = \arg\max(\mathbf{S}_{kt} + \mathbf{S}_{k-1,t-1}, \mathbf{S}_{k,t-1})$
6. end
7. end
8. $k = K, t = T$
9. repeat
10. if $\mathbf{\Phi}_{kt} == 1$
11. then $z_{kt} = 1,\ t = t - 1,\ k = k - 1$
12. else $t = t - 1$
13. until $k == 0$

Abbildung A.2.: Pseudocode für den Alignmentschritt der FAMs.

1. Berechne Distanzmatrix $\mathbf{D}$
2. for $k = 1..K$
3. for $t = 1..T$
4. $\mathbf{D}_{kt} = \min(\mathbf{D}_{kt} + \mathbf{D}_{k,t-1}, \mathbf{D}_{k-1,t})$
5. $\mathbf{\Phi}_{kt} = \arg\min(\mathbf{D}_{kt} + \mathbf{D}_{k,t-1}, \mathbf{D}_{k-1,t})$
6. end
7. end
8. $k = \min_{k'} \mathbf{D}_{k',t},\ t = T$
9. repeat
10. if $\mathbf{\Phi}_{kt} == 1$
11. then $z_{kt} = 1,\ t = t - 1$
12. else $k = k - 1$
13. until $t == 0$

Abbildung A.3.: Pseudocode für den Alignmentschritt von *OMMbest*.

1. Berechne Distanzmatrizen $\mathbf{D},\ \mathbf{D}^{\alpha} = \mathbf{D},\ \mathbf{D}^{\beta} = rotate180(\mathbf{D})$
2. for $k = 1..K$
3. for $t = 1..T$
4. $a11 = \mathbf{D}^{\alpha}_{k-1,t},\ b11 = \mathbf{D}^{\beta}_{k-1,t}$
5. $a12 = \mathbf{D}^{\alpha}_{k,t} + \mathbf{D}^{\alpha}_{k,t-1},\ b12 = \mathbf{D}^{\beta}_{kt} + \mathbf{D}^{\beta}_{k,t-1}$
6. $\mathbf{D}^{\alpha}_{kt} = a11 + \log(1 + \exp(a12 - a11))$
7. $\mathbf{D}^{\beta}_{kt} = b11 + \log(1 + \exp(b12 - b11))$
8. end
9. end
10. $\alpha = \mathbf{D}^{\alpha},\ \beta = rotate180(\mathbf{D}^{\beta})$
11. for $k = 1..K$
12. for $t = 1..T$
13. $r_{kt} = \exp(\mathbf{D}^{\alpha}_{k,t-1} + \mathbf{D}_{kt} + \mathbf{D}^{\beta}_{k,t-1} - \mathbf{D}^{\alpha}_{K,T})$
14. end
15. end

Abbildung A.4.: Pseudocode für den Alignmentschritt von *OMMall*.

A.3. Komplexitätsanalyse

Um die Notation zu vereinfachen, arbeiten wir in diesem Abschnitt mit gleichen Sequenzlängen $|\mathbf{S}_1| = .. = |\mathbf{S}_N| = T$. Für die Komplexitätsbetrachtungen des Trainingsschritts kann man außerdem die N_{y_i} Trainingsbeispiele aller $y_i \in Y$ Klassen zu einer Menge von N Sequenzen zusammenfassen.

A.3.1. OMM

Der Alignmentschritt der OMMs erfordert die sequenzweise Berechnung des optimalen Pfades (*OMMbest*) bzw. der Produktdichte (*OMMall*). Die Erstellung der Distanzmatrix schlägt mit $O(K \cdot d \cdot T)$, das dynamische Programm für beide Methoden mit $O(K \cdot T)$ und die Rückverzeigerung für *OMMbest* mit $O(K)$ zu Buche. Insgesamt ergibt sich somit ein Rechenaufwand von $O_{Al}(N \cdot T \cdot K \cdot d)$.

Die Zeitkomplexität der Regression hängt von der Prototypgröße K, der Anzahl der Sequenzen N, deren Länge T und Dimensionalität d multiplikativ ab. Insgesamt erhalten wir wiederum $O_{Re}(N \cdot T \cdot K \cdot d)$, was schließlich multipliziert mit der Anzahl der Iterationen

die Gesamtkomplexität des Trainings darstellt.

Für die Klassifikation muss jede der N_{Test} Testsequenzen gegen den Prototypen jeder Klasse aligniert werden um die Produktionswahrscheinlichkeit zu erhalten, was einen Aufwand von $O(N_{Test} \cdot T \cdot M \cdot K \cdot d)$ verursacht. Die Komplexität des *OMMKDE* berechnet sich für N_{Test} Testsequenzen und N Trainings- bzw. Dichteschätzungssequenzen zu $O(N \cdot N_{Test} \cdot T^2 \cdot d)$, womit das Verfahren für kleine N bzw. N_{Test} prädestiniert ist, da in dem Fall parametrische Verfahren oft an Unterrepräsentation bzgl. der Daten leiden.

A.3.2. FAM

Im Alignment-Schritt muss pro Sequenz und Folgenlänge

- die Folge überlappender Sequenzabschnitte gebildet: $O(L^j)$,
- diese normiert: $O(T \cdot d \cdot L^j)$,
- das Skalarprodukt (als Matrixprodukt) berechnet: $O(K^j \cdot d \cdot L^j \cdot T)$,
- ein dynamisches Programm der Größe $K^j \times T$ evaluiert: $O(K^j \cdot T)$,
- und die Rückverzeigerung in $O(K^j)$ vollzogen werden.

Aus der Vereinfachung von

$$O\left(N \cdot \sum_{j=1}^{J} \left(O(L^j) + O(T \cdot d \cdot L^j) + O(K^j \cdot T \cdot d \cdot L^j) + O(K^j \cdot T) + O(K^j)\right)\right)$$

zu $O(N \cdot \sum_{j=1}^{J} O(K^j \cdot T \cdot d \cdot L^j))$ können wir durch Einsetzen von $D = K^j \cdot L^j$ den zeitlichen Aufwand $O_{Al}(N \cdot J \cdot D \cdot d \cdot T)$ ableiten. Wie zu erkennen ist, sollte die Anzahl der verwendeten Folgenlängen J und das D möglichst klein gewählt werden. Für die diskriminative *FASVM* ist zu beachten, dass M 2-Klassen-Probleme vorliegen, d.h. der Faktor M zu multiplizieren ist.

Bei der Regression unterscheidet sich der Zeitbedarf je nach eingesetzter Methode. Für die einfache Mittelwertschätzung des *FAMmean* werden je Sequenz $O(d \cdot D \cdot J)$, also insgesamt $O_{FMAmean}(N \cdot J \cdot D \cdot d)$ Operationen benötigt.

Bei den *FASVMs* lässt sich die Komplexität des Regressionsschritts aufgrund diverser in der *libsvm* integrierter Optimierungsverfahren nur grob mit $O(N^2 \cdot d \cdot D \cdot J)$ angeben. Die diskriminative Variante unterscheidet sich dahingehend nicht, jedoch sind je nach Größe der zweiten Klasse hier u.U. sehr viel mehr Trainingsbeispiele vorhanden.

Die Klassifikation von N_{Test} Testsequenzen beträgt für alle Varianten $O(N_{Test} \cdot M \cdot J \cdot D \cdot d \cdot T) = O_{Al}$, da wiederum alle Sequenzen gegen die Prototypen aligniert werden müssen. Die Berechnung der Distanz zum Mittelwert für *FAMmean* sowie des Skalarprodukts aus Sequenzmerkmalsvektor und Prototyp bei den SVM-Versionen ist dabei mit $O(N_{Test} \cdot J \cdot d \cdot D)$ enthalten und spielt aufgrund der effizienten Implementation des Matrixprodukts in Matlab eine eher untergeordnete Rolle.

A.4. Toolboxen

Für jedes der in Kapitel 5 und 6 vorgestellten Verfahren entwickelten wir eine einfach zu verwendende Matlab-Toolbox. Einige Funktionen werden von beiden Verfahren verwendet. Die Beschreibung erfolgt für diese nur einmal.

A.4.1. Ordered-Means-Modelle Toolbox

Diese Toolbox implementiert die in Kapitel 5 vorgestellten Hauptverfahren *OMMbest* und *OMMall*. Dabei sind die beiden Varianten aus Übersichtsgründen getrennt voneinander realisiert und gemeinsame Methoden ausgelagert sowie rechenintensive Abschnitte in C programmiert worden. Die Funktionsarten gliedern sich hierarchisch in Validierungs-, Trainings- und Test- sowie Initialisierungs-, Alignment- und Regressionsfunktionen.

fast_dpsur

```
[Trace, Dcum] = fast_dpsur(D)
```

Diese Funktion berechnet innerhalb des Alignmentsschritts aus der Distanzmatrix `D` die kumulative Distanzmatrix `Dcum` und die Tracebackmatrix `Trace`. Dieser Vorgang ist sehr rechenintensiv und wurde von uns daher in C implementiert.

omm_init

```
[Mu_init, multi_data] = omm_init(options, S, K)
```

`omm_init` erzeugt in Abhängigkeit der verwendeten Initialisierungsvariante, welche im `options`-struct spezifiziert ist, anhand der Daten in `S` und der vorgegebenen Prototypgröße `K` initiale Prototypen (Referenzvektoren) `Mu_init`. Sollte dabei ein multiples Alignment (ebenfalls in `options`) als Grundlage herangezogen werden, so sind in `multi_data` die je nach Merkmalsvektortyp transformierten multivariaten Sequenzen enthalten.

ommbest_align

```
[Zall, Dall] = ommbest_align(options, S, Mu)
```

Die Implementierung des Alignmentschritts berechnet anhand der N Daten in `S` und dem Prototypen `Mu` die als cell-array von N Matrizen organisierten, diskreten Zuordnungen `Zall` sowie die minimale Distanzsumme `Dall`.

ommbest_reg

```
[Mu_new, alpha] = ommbest_reg(options, S, Z, lambda)
```

Anhand der in `ommbest_align` berechneten Zuordnungen `Z` und der Daten in `S` sowie im regularisierten Fall unter Berücksichtigung von `lambda` werden die neuen Prototypen `Mu_new` sowie der geschätzte Skalierungsfaktor des Einheitsvektors `alpha` gemäß Gleichung 5.53 berechnet.

ommbest_calc_error

```
E = ommbest_calc_error(options, S, Mu, alpha, Z, lambda)
```

Diese Funktion berechnet den Wert der Fehlerfunktion anhand der Daten in `S`, der aktuellen Prototypen in `Mu` und den diskreten Zuordnungen in `Z`. Im regularisierten Fall werden zudem der Penaltyfaktor `lambda` und der Skalierungsfaktor des Einheitsvektors `alpha` gemäß Gleichung 5.50 berücksichtigt.

ommbest_train

```
[Mu, alpha, info] = ommbest_train(options, S, Mu_init, lambda)
```

Die Funktion `ommbest_train` trainiert aus den Daten in `S` unter Berücksichtigung von `lambda` ein deterministisches OMM. Das Argument `Mu_init` ist die Initalisierung der Prototypen wie sie durch `omm_init` berechnet wird. Die Rückgabewerte der Funktion sind zum einen die trainierten Prototypen in `Mu`, die geschätzten `alpha` der Regression sowie ein `info`-Struct mit Information zum Verlauf der Fehlerfunktion und den Zuordnungen des Alignments. `ommbest_train` arbeitet nach dem in Abschnitt 5.4 beschriebenen iterativen Projektions-Regressions-Schema.

ommbest_test

```
[class, dist] = ommbest_test(options, S, Mu)
```

`ommbest_test` klassifiziert ungelabelte Testsequenzen des Structs `S` anhand der minimalen Distanzsumme zu den Prototypen `Mu`. Die Rückgabewerte sind die Klassenlabel `class` und eine Distanzsummenmatrix `dist`, die für jede Testsequenz die Distanzsummen zu den Prototypen notiert.

ommbest_cv

```
[perf, info, folds] = ommbest_cv(options, data, folds)
```

Diese Funktion führt eine Kreuzvalidierung auf den Daten `data` durch. In `options` werden alle Optionen spezifiziert, der Hyperparametersuchraum, die Anzahl der Folds sowie Logdateien und weitere Verfahrensparameter. Als optionales Argument kann `ommbest_cv` ein struct `folds` übergeben werden, in dem die Daten nach Folds partitioniert sind. Wird darauf verzichtet, partitioniert `ommbest_cv` die Daten zufällig selbst. Die Rückgabewerte sind eine Matrix `perf`, die die Performanz der Hyperparameter aus `options` notiert, `info`, eine detailliertere Ergebnisaufschlüsselung und `folds`, ein struct, das die Folds der Hyperparametersuche umfasst. Dieses kann für vergleichbare Analysen zusätzlicher Hyperparameter herangezogen werden.

ommall_align

```
[Rall, logP, Dall] = ommall_align(options, S, Mu, sigma)
```

Wie `ommbest_align`, jedoch mit den Responsilibities `Rall` statt diskreter Zuordnungen, den Logarithmen der Produktionswahrscheinlichkeiten sowie dem zusätzlichen Varianzparameter `sigma` der Emissionsdichten.

ommall_reg
```
[Mu_new, sigma_est] = ommall_reg(options, S, R)
```

Wie `ommbest_reg`, jedoch mit den Responsilibities `R` statt diskreter Zuordnungen.

ommall_calc_error
```
E = ommall_calc_error(options, P)
```

Der Wert der Fehlerfunktion wird allein anhand der Summe der negativen logarithmischen Produktionswahrscheinlichkeiten aus dem aktuellen Alignmentschritt errechnet.

ommall_train
```
[Mu, info] = ommall_train(options, S, Mu_init, sigma)
```

Wie `ommbest_train`, jedoch mit dem Varianzparameter `sigma` der Emissionsdichten und probalistischen OMM-Prototypen als Ergebnis.

ommall_test
```
[class, dist] = ommall_test(options, S, Mu, sigma)
```

Wie `ommbest_test`, jedoch mit dem zusätzlichen Varianzparameter `sigma` der Emissionsdichten. Getestet wird mit den zur Distanzsumme proportionalen, logarithmischen Produktionswahrscheinlichkeiten.

ommall_cv
```
[perf, info] = ommall_cv(options, data, folds)
```

Wie `ommbest_cv`.

mk_omm_options
```
options = mk_omm_options()
```

Die Funktion `mk_omm_options` erzeugt ein mit Defaultwerten belegtes `options`-struct.

A.4.2. Feature-Alignment-Maschinen Toolbox

Diese Toolbox realisiert die in dieser Diplomarbeit vorgestellten FAM-Algorithmen. Zu unterscheiden ist zwischen Funktionen, die eine Hyperparametersuche ermöglichen (`fam_crossvalid` und `fam_hpsearch`), Funktionen, die das Training der verschiedenen Algorithmen durchführen (`fam_train_single`, `fam_train_disc`, `fam_train_multi`, `fam_fsmean`), Testfunktionen (`fam_test_single`, `fam_test_fsmean`), die unbekannte Sequenzen anhand trainierter Prototypen klassifizieren, Funktionen, die aufgrund ihrer Komplexität von uns ausgelagert wurden (`fam_align`, `fam_reg`, `fam_reg_single`), sowie nützlichen Zusatzfunktionen (`fam_norm_data`, `fam_init`, `mk_fam_options`). Darüberhinaus wurde das rechenintensive dynamische Programm des Alignments in C implementiert und findet sich in der Funktion `fast_dpbij`.

fam_crossvalid

```
[perf, results, folds] = fam_crossvalid(options, data, folds)
```

Diese Funktion führt eine Kreuzvalidierung auf den Daten `data` durch. In `options` werden alle Optionen spezifiziert, der Hyperparametersuchraum, die Anzahl der Folds, welcher Algorithmus verwendet werden soll sowie Logdateien und weitere Verfahrensparameter. Als optionales Argument kann `fam_crossvalid` ein struct `folds` übergeben werden, in dem die Daten nach Folds partitioniert sind. Wird darauf verzichtet, partitioniert `fam_crossvalid` die Daten zufällig selbst. Die Rückgabewerte sind eine Matrix `perf`, die die Performanz der Hyperparameter aus `options` notiert, `results`, eine detailliertere Ergebnissaufschlüsselung, `folds`, ein struct, das die Folds der Hyperparametersuche umfasst. Dieses kann für vergleichbare Analysen zusätzlicher Hyperparameter herangezogen werden.

fam_hpsearch

```
[perf, results] = fam_hpsearch(options, train, test)
```

Alternativ zur Kreuzvalidierung können mit `fam_hpsearch` optimale Hyperparameter für ein Testset `test` auf einem Trainingsset `train` gefunden werden. Die Rückgabewerte sind analog zu denen von `fam_crossvalid`.

fam_train_single

```
[W, o] = fam_train_single(options, data, W, D, nu)
```

Die Funktion `fam_train_single` trainiert aus den Daten aus `data` unter Berücksichtigung von `D` und `nu` eine Ein-Klassen-*FASVM*. Das Argument `W` ist die Initialisierung des Prototyp $\mathbf{w}$, wie sie beispielsweise durch `fam_init` berechnet wird. Die Rückgabewerte der Funktion sind der trainierte Prototyp `W` und der Verlauf der Zielfunktion `o`. `fam_train_single` arbeitet nach dem in Abschnitt 6.7 beschriebenen iterativen Projektions-Regressions-Schema.

fam_train_disc

```
[W, o] = fam_train_disc(options, data, W, D, C)
```

`fam_train_disc` realisiert das diskriminative binäre Training einer *FASVM*. Die Argumente und Rückgabewerte sind analog zu `fam_train_single`.

fam_train_multi

```
[W, o] = fam_train_multi(options, data, W, D, C)
```

Die Funktion `fam_train_multi` trainiert den diskriminativen Multiklassenfall einer *FASVM* nach dem one-against-all Prinzip. Die Argumente und Rückgabewerte sind analog zu `fam_train_single`.

fam_train_fsmean

```
[MeanX, o] = fam_train_fsmean(opt, data, X_init, D)
```

`fam_train_fsmean` realisiert den *FAMmean* zur Klassifikation, indem für jede Klasse aus `data` ein eigener Prototyp berechnet wird. Die Rückgabe `MeanX` ist ein struct, dass diese Prototypen repräsentiert.

fam_test

```
[class, score, scores] = fam_test_single(options, data, W, b, D)
```

Die Funktion `fam_test` klassifiziert die Sequenzen in `data` nach maximaler Ähnlichkeit zu den Prototypen im struct `W`. `D` ist der zu `W` gehörende Hyperparameter D. Die Rückgabewerte sind die Klassenlabel `class`, die maximalen Scores `score`, die zu diesen Klassen gehören, sowie alle Skalarprodukte `scores` jeder Sequenz aus `data` zu allen `W`, anhand derer die Klasse bestimmt wurde.

fam_align

```
[X, Z] = fam_align(options, Sin, W, sv_list, D)
```

`fam_align` realisiert das Alignment. Die Sequenzen `Sin` werden gegen den Prototyp `W` aligniert. `sv_list` ist ein Indexvektor, korrespondierend zu den Sequenzen aus `Sin`, der anzeigt, ob eine Sequenz aligniert werden soll oder nicht (vgl. Abschnitt 6.7). `D` steht für der Hyperparameter D und ist eine natürliche Zahl. Das `options`-struct enthält alle weiteren Parameter und Informationen, beispielsweise den für das Alignment benötigten Verfahrensparameter L.

fast_dpbij

```
[Trace, K] = fast_dpbij(P)
```

Diese Funktion berechnet aus der Skalarproduktmatrix `P` die kumulative Skalarproduktmatrix `K` und die Tracebackmatrix `Trace`. Dieser Vorgang ist sehr rechenintensiv und wurde von uns daher in C implementiert.

fam_reg_disc

```
[V, b, obj, sv_list, nsv, Alpha] = fam_reg(options, X, C)
```

`fam_reg_disc` berechnet aus den Merkmalsraumprojektionen `X` der Sequenzen die diskriminative Regression mit Hilfe einer C-SVM. Als Argumente erwartet diese Funktion `X`, die Merkmalsraumrepräsentanten der Sequenzen, und `C`, das Gewicht des Penaltys als Parameter für die SVM. Rückgaben sind `V`, der Gewichtsvektor der SVM, `b`, der Bias, `obj`, der Wert der Zielfunktion, `sv_list`, ein Indexvektor, der anzeigt, welche Vektoren aus `X` die Gleichung 6.17 erfüllen, `nsv`, die Anzahl dieser Vektoren und `Alpha`, die α_i der dualen Zielfunktion der SVM.

fam_reg_single

```
[V, obj, sv_list, nsv, Alpha] = fam_reg_single(options, X, nu)
```

Diese Funktion realisiert die Regression für die Ein-Klassen-*FASVM*. Die Parameter und Rückgabewerte sind analog zu `fam_reg_disc`, nur dass auf die Rückgabe von `b` verzichtet werden kann.

mk_fam_options

```
options = mk_fam_options()
```

Die Funktion `mk_fam_options` erzeugt ein mit Defaultwerten belegtes `options`-struct.

fam_init

```
[W_init, false_bias, dataout] = fam_init(options, data, D, short_behavior)
```

`fam_init` führt diverse Initalisierungen durch. Zunächst werden die Prototypen `W_init` wie in Abschnitt 6.7 vorbelegt. Zusätzlich werden die Sequenzen `data` in Abhängigkeit von `D` und `short_behavior` auf minimale Länge gepaddet (vgl. Abschnitt 6.2) oder entfernt. `false_bias` ist die Anzahl der entfernten Sequenzen, `dataout` die vorverarbeiteten Sequenzen.

fam_fill_seq

```
Sout = fam_fill_seq(S, T, fill_mode, fill_vec)
```

Diese Funktion realisiert das Padding der Sequenz `S` auf Länge `T` in Abhängigkeit des in `fil_mode` beschriebenen Auffüllmodus. Mögliche Vorgehensweisen sind das Auffüllen mit einem sequenzspezifischen Mittelwert (`fill_mode = 1`), das Wiederholen des ersten und des letzten Sequenzvektors (`fill_mode = 2`) sowie das Auffüllen mit dem optionalen Vektor `fill_vec` (`fill_mode = 3`).

fam_norm_data

```
ndata = fam_norm_data(data)
```

Diese Funktion normiert die Sequenzenvektoren aus `data` auf euklidische Länge 1.

B. Notation

	Allgemeines
a	Skalar bzw. Variable
$\mathbf{x}$	Vektor
$\mathbf{W}$	Matrix
$\|\mathbf{x}\|$	L2-Norm eines Vektors (euklidische Norm)
$p(\cdot)$	Wahrscheinlichkeitsdichte
$P(\cdot)$	Wahrscheinlichkeit
d	Dimensionalität der Sequenz/Zeitserie (Eingaberaum)
$\mathbb{R}^d$	d-dimensionaler Raum der reellen Zahlen
$\mathbf{x}$	Merkmalsvektor aus $\mathbb{R}^d$
$\mathbf{S}$	multidimensionale Sequenz bzw. Zeitserie
T	Länge der Sequenz bzw. Zeitserie $\mathbf{S}$
$\mathbf{s}_t$	Sequenz- bzw. Signalvektor zum Zeitpunkt $t = 1, .., T$
Y	Menge der Klassen
y_i	Klassenzuordnung (Label) von $\mathbf{x}^i$ bzw. $\mathbf{S}^i$
M	Anzahl der Klassen (Kategorien) $\|Y\|$
N_{y_i}	Anzahl der Sequenzen/Daten in Klasse $y_i \in Y$
k	Anzahl der Folds bei der Kreuzvalidierung (*k-fold*)
	Hidden-Markov-Modelle (HMM)
Λ	Hidden-Markov-Modell (HMM)
$\boldsymbol{\pi}$	Vektor der Anfangswahrscheinlichkeiten
$\mathbf{A}$	Matrix der Übergangswahrscheinlichkeiten
$a_{k'k}$	Übergangswahrscheinlichkeit von Zustand k' nach k
$\mathbf{B}$	Matrix der Emissionswahrscheinlichkeiten
$b_k(\mathbf{s}_t)$	Emissionswahrscheinlichkeit von $\mathbf{s}_t$ in Zustand k
	Support-Vektor-Maschinen (SVM)
$\mathbf{w}$	Normalenvektor der Hyperebene
b	Bias = Abstand der Hyperebene vom Ursprung
C	Hyperparameter (Strafmaß) der C-SVM
$\boldsymbol{\alpha}$	Vektor der Lagrange-Multiplikatoren der dualen Zielfunktion
ξ_i	Hilfsvariable für C-SVM (*slack variable*)
$\Phi(.)$	Transformation in den erweiterter Merkmalsraum
$k(.)$	Kernfunktion
ν	Hyperparameter (Anteil der Supportvektoren) der Ein-Klassen-SVM
ρ	Abstand der Hyperebene der Ein-Klassen-SVM vom Ursprung

	Ordered-Means-Modelle (OMM)
Ω	Ordered-Means-Modell (OMM)
K	Modellgröße, Anzahl der Referenzvektoren bzw. Zustände
$\mathbf{W}$	Matrix der K Mittelwertvektoren
$\boldsymbol{\mu}$	Mittelwert-/Modellvektor
$\mathbf{q}$	Pfad von Zuständen durch das Modell
q_t	Zustand zum Zeitpunkt t
$b_k(\mathbf{s}_t)$	Emissionswahrscheinlichkeit von $\mathbf{s}_t$ in Zustand k
$h^i_{\mathbf{q}}$	Pfadwahrscheinlichkeit von Pfad $\mathbf{q}$ zu Sequenz $\mathbf{S}^i$
$\mathbf{R}$	Matrix der Verantwortlichkeiten (*OMMall*)
r^i_{kt}	Verantwortlichkeit von $\mathbf{s}^i_t$ für $\boldsymbol{\mu}_k$
σ	Varianzparameter der Emissionsdichten
$\mathbf{Z}$	Matrix der Zuordnungen von $\mathbf{S}$ auf $\mathbf{W}$ (*OMMbest*)
z^i_{kt}	Zuordnung von $\mathbf{s}^i_t$ aus $\mathbf{S}^i$ auf $\boldsymbol{\mu}_k$
λ	Regularisierungsparameter
	Feature-Alignment-Maschinen (FAM)
$\mathbf{x}$	Merkmalsvektor einer Sequenz nach der Transformation
$\bar{\mathbf{S}}$	Folge überlappender Sequenzabschnitte der Sequenz $\mathbf{S}$
$\bar{\mathbf{s}}_t$	Sequenzmustervektor zum Zeitpunkt t
$\mathbf{w}$	Vektor von Referenzvektoren
$\mathbf{W}^j$	Matrix der K^j Referenzvektoren $\mathbf{w}^j$ für L^j
$\bar{\mathbf{w}}^j_k$	k-ter Referenzmustervektor der Folgenlänge L^j
J	Anzahl der benutzten Folgenlängen
L	Vektor der benutzten Folgenlängen $L = \{L^1, .., L^J\}$
$\mathbf{Z}^j$	Matrix der Zuordnungen von $\bar{\mathbf{S}}$ auf alle $\bar{\mathbf{w}}^j$
z_{kt}	Zuordnung von $\bar{\mathbf{s}}_t$ aus $\bar{\mathbf{S}}$ auf $\bar{\mathbf{w}}_k$

Literaturverzeichnis

[AHB04] A. Andreeva, D. Howorth, S. E. Brenner, T. J. P. Hubbard, C. Chothia, A. G. Murzin. *SCOP database in 2004: refinements integrate structure and sequence family data*. Acid Research, Vol. 32, pp. D226-D229, 2004.

[AL90] S. F. Altschul, W. Gish, W. Miller, E. Myers, D. Lipman. *Basic Local Alignment Search Tool*. Journal of Molecular Biology 215, pp. 403-410, 1990.

[Bir01] E. Birney. *Hidden Markov models in biological sequence analysis*. IBM J. Res. & Dev., Vol. 45, Nr. 3/4, 2001. `www.research.ibm.com/journal/rd/453/birney.pdf`

[Bur98] C. J. C. Burges. *A Tutorial on Support Vector Machines for Pattern Recognition*. Data Mining and Knowledge Discovery, Vol. 2, Nr. 2, pp. 121-167, 1998.

[Bus04] S. Busuttil, J. Abela, G. J. Pace. *Support Vector Machines with Profile-Based Kernels for Remote Protein Homology Detection*. Genome Informatics, Vol. 15, Nr. 2, pp. 191-200, 2004.

[Cha04] C. C. Chang, C. J. Lin. *LIBSVM: a Library for Support Vector Machines*. `http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf`

[CKHP02] S. Chu, E. Keogh, D. Hart, M. Pazzani. *Iterative deepening dynamic time warping for time series*. In: Proc 2 na SIAM International Conference on Data Mining. 2002.

[CM98] V. Cherkassky and F. Mulier. *Learning from Data – Concepts, Theory and Methods*. John Wiley & Sons, New York, 1998.

[CS00] N. Cristianini, J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

[CTZ04] K. H. Choo, J. C. Tong, L. Zhang. *Recent applications of Hidden Markov Models in computational biology*. Genomics Proteomics Bioinformatics, Vol. 2, Nr. 2, pp. 84-96, 2004.

[DHS01] R. O. Duda, P. E. Hart, D. Stork. *Pattern Classification*. WileyInterscience, 2001.

[DLR77] A. Dempster, N. Laird, D. Rubin. *Maximum likelihood from incomplete data via the em algorithm*. Journal of the Royal Statistical Society, Series B, Vol. 39, Nr. 1, pp. 1-38, 1977.

[Duo04] T. Doung. *Bandwidth selectors for multivariate kernel density estimation*. PhD thesis, School of Mathematics and Statistics, University of Western Australia, 2004.

[Dur98] R. Durbin, S. R. Eddy, A. Krogh, G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[Edd96] S. R. Eddy. *Hidden Markov models*. Current Opinion in Structural Biology, Vol. 6, pp. 361-365, 1996.

[Edd98] S. R. Eddy. *Profile hidden Markov models* Bioinformatics, Vol. 14, pp. 755-763, 1998.

[Fre77] M. J. Freyer. *A review of Some Non-parametric Methods of Density Estimation*. Journal of the Institute of Mathematicas and its Applications, Vol. 20, pp. 335-354, 1977.

[Gie02] R. Giegerich. *Sequence Similarity and Dynamic Programming*. Lecture Notes Bioinformatics Summerschool, Bad Urach 2002.

[Gor03] L. Gordon, A. Y. Chervonenkis, A. J. Gammerman, I. A. Shahmuradov, V. V. Solovyev. *Sequence alignment kernel for recognition of promoter regions*. Bioinformatics, Vol. 19, Nr. 15, pp. 1964-1971, 2003.

[Gou01] J. Gough, K. Karplus, R. Hughey, C. Chothia. *Assignment of homology to genome sequences using a library of Hidden Markov Models that represent all proteins of known structure*. Journal of Molecular Biology, Vol. 31, pp. 903-919, 2001.

[Guy05] I. Guyon. *SVM Application List*. `http://www.clopinet.com/isabelle/Projects/SVM/applist.html`

[Han01] A. Hansen. *Bioinformatik: ein Leitfaden für Naturwissenschaftler*. Birkhäuser, Basel, 2001.

[HH92] S. Henikoff, J. Henikoff. *Amino acid substitution matrices from protein blocks*. Proceedings of National Academy Science of the USA, Vol. 89, pp. 10915-10919, 1992.

[Hol98] I. Holmes, R. Durbin. *Dynamic programming alignment accuracy*. Journal of computational biology, Vol. 5, Nr. 13, pp. 493-504, 1998. `http://portal.acm.org/ft_gateway.cfm?id=279102&type=pdf`

[Hou03] Y. Hou, W. Hsu, M. L. Lee, C. Bystroff. *Efficient remote homology detection using local structure*. Bioinformatics Vol. 19, Nr. 17, pp. 2294-2301, 2003.

[HTF01] T. Hastie, R. Tibshirani, J. Friedman. *The Elements of Statistical Learning*. Springer, New York, 2001.

[HTG94] D. Higgins, J. Thompson, T. Gibson. *CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice*. Nucleic Acids Research, Vol. 22, pp. 4673-4680. 1994.

[Jaa98] T. Jaakkola, M. Diekhans, D. Haussler. *A discriminative framework for detecting remote protein homologies*. `http://www.cse.ucsc.edu/research/compbio/research.html`

[Jaa99] T. Jaakkola, M. Diekhans, D. Haussler. *Using the Fisher kernel method to detect remote protein homologies*. 7th Intelligent Systems in Molecular Biology, pp. 149-158, 1999.

[Joa99] T. Joachims. *Making large-Scale SVM Learning Practical*. In: Advances in Kernel Methods - Support Vector Learning, B. Schlkopf and C. Burges and A. Smola (ed.), MIT Press, 1999.

[Koh97] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berlin, 2nd edition, 1997.

[Kos01] T. Koski. *Hidden Markov Models for Bioinformatics*. Kluwer Academic Publishers, Dordrecht, 2001.

[KP99] E. J. Keogh, M. J. Pazzani. *Scaling up Dynamic Time Warping to Massive Datasets*. Proc. Principles and Practice of Knowledge Discovery in Databases, 1999.

[KR71] N. G. Kingsbury, P. J. W. Rayner. *Digital Filtering Using Logarithmic Arithmetic* Electronics Letters, Vol. 7, pp. 56-58, 1971.

[Kro93] A. Krogh, M. Brown, I. S. Mian, K. Sjolander, D. Haussler. *Hidden Markov Models in Computational Biology: Applications to Protein Modeling*. Journal of Molecular Biology, Vol. 23, Nr. 5, pp. 1501-1531, 1993.

[Kro94] A. Krogh. *Hidden Markov models for labeled sequences*. Proceedings of the 12th IAPR International Conference on Pattern Recognition, pp. 140-144, Los Alamitos, 1994.

[Lai02] L. Liao, W. S. Noble. *Combining pairwise sequence similarity and support vector machines for remote protein homology detection*. Proceedings of the Sixth Annual International Conference on Research in Computational Molecular Biology, pp. 225-232, 2002.

[Lan96] P. Langley. *Elements of Machine Learning*. Morgan Kaufmann, Palo Alto, 1996.

[LBH02] L. Lo Conte, S. E. Brenner, T. J. P. Hubbard, C. Chothia, A. G. Murzin. *SCOP database in 2002: refinements accommodate structural genomics*. Nucleic Acid Research, Vol. 30, Nr. 1, pp. 264-267, 2002.

[Les04] C. Leslie, E. Eskin, A. Cohen, J. Weston, W. S. Noble. *Mismatch string kernels for discriminative protein classification*. Bioinformatics Advance Access online. `http://bioinformatics.oupjournals.org/cgi/reprint/btg431v1.pdf`

[Lin05] LIBSVM. `www.csie.ntu.edu.tw/~cjlin/libsvm/`

[Mar03] F. Markowetz, L. Edler, M. Vingron. *Support Vector Machines for Protein Fold Class Prediction*. Biometrical Journal, Vol. 45, Nr. 3, pp. 377-389, 2003.

[Mat05] *The MathWorks - MATLAB and Simulink for Technical Computing*. `www.mathworks.com`

[MBH95] A. G. Murzin, S. E. Brenner, T. J. P. Hubbard, C. Chothia. *SCOP: a structural classification of proteins database for the investigation of sequences and structures*. Journal of Molecular Biology, Vol. 24, Nr. 7, pp. 536-540, 1995.

[Mei05] P. Meinicke, S. Klanke, R. Memisevic, H. Ritter. *Principal Surfaces from Unsupervised Kernel Regression*. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 27, No. 9, pp. 1379-1391, 2005.

[Mit97] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[ML81] J. V. Maizel, R. P. Lenk. *Enhanced graphic matrix analysis of nucleic acid and protein sequences*. Proceedings of National Academy Science of the USA Vol. 78, pp. 7665-7669, 1981.

[Mül03] H. M. Müller. *Neurobiologische Grundlagen der Sprachfähigkeit*. In: G. Rickheit, T. Herrmann, W. Deutsch (Hrsg.) Psycholinguistik: Ein internationales Handbuch. de Gruyter, Berlin, pp. 57-80, 2003.

[Mur94] K. P. Murphy. *Biological Sequence Comparison: An Overview of Techniques*. `http://iteseer.ist.psu.edu/murphy94biological.html`

[MZA04] J. Ma, Y. Zhao, S. Ahalt. *OSU SVM Classifier Matlab Toolbox*. `www.ece.osu.edu/~maj/osu_svm/`

[Nil96] N. J. Nilsson. *Introduction to Machine Learning*. Unveröffentlicht. `http://ai.stanford.edu/people/nilsson/mlbook.html`

[NW70] S. B. Needleman, C. D. Wunsch. *A general method applicable to the search for similarities in the aminoacid sequence of two proteins*. Journal of Molecular Biology, Vol. 48, pp. 433-453, 1970.

[Pla98] J. Platt. *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*. Microsoft Research Technical Report MSR-TR-98-14, 1998.

[PF04] T. Plötz, G. A. Fink. *Feature extraction for improved Profile HMM based biological sequence analysis*. In Proceedings of the International Conference on Pattern Recognition, Nr. 2, pp. 315-318, IEEE, 2004.

[PF05] T. Plötz, G. Fink. *A new approch for HMM based protein sequence family modeling and its application to remote homology classification*. In Proceedings of the Workshop on Statistical Signal Processing, Bordeaux, France, 2005.

[PL88] W. R. Pearson, D. J. Lipman. *Improved tools for biological sequence comparison*. Proceedings of National Academy Science of the USA, Vol. 85, pp. 2444-2448, 1988.

[Plö05] T. Plötz. *Advanced Stochastic Protein Sequence Analysis*. Universität Bielefeld, Technische Fakultät, Dissertation, 2005. `http://bieson.ub.uni-bielefeld.de/volltexte/2005/718/`

[Rab89] L. R. Rabiner. *A tutorial on hidden Markov models and selected applications in speech recognition*. Proceedings of the IEEE, Vol. 77, Nr. 2, pp. 257-286, 1989.

[Rat03] T. M. Rath, R. Manmatha. *Lower-Bounding of Dynamic Time Warping Distances for Multivariate Time Series*. Center for Intelligent Information Retrieval, University of Massachusetts, Technical Report MM-40, 2003. `http://ciir.cs.umass.edu/pubfiles/mm-40.pdf`

[Rau01] R. Rauhut. *Biniformatik*. Wiley-VCH, Weinheim, 2001.

[Sai04] H. Saigo, J.-P. Vert, N. Ueda, T. Akutsu. *Protein homology detection using string alignment kernels*. Bioinformatics, Vol. 20, Nr. 11, pp. 1682-1689, 2004.

[SAM95] R. Hughey, A. Krogh. *SAM : Sequence alignment and modeling software system*. Technical Report UCSC-CRL-95-7, University of California, Santa Cruz, 1995.

[SC78] H. Sakoe, S. Chiba. *Dynamic programming algorithm optimization for spoken word recognition*. IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP 26, pp. 43-49, 1978.

[SD96] S. Qian, D. Chen. *Joint Time-Frequency Analysis – Methods and Applications*. Prentice Hall, 1996.

[SED97] E. L. Sonnhammer, S. R. Eddy, R. Durbin. *Pfam: a comprehensive database of protein families based on seed alignments*. Proteins, Vol. 28, pp. 405-420, 1997.

[Sil86] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London, 1986.

[SK91] J. Schürmann, U. Kreel. *Neuronale Netze in der Mustererkennung: Multilayer-Perzeptron und Polynomklassifikator*. In: HMD, Vol. 159, pp. 110-122, 1991.

[SS02] B. Schölkopf, A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[Ste56] C. Stein. *Inadmissibility of the Usual Estimator for the Mean of a Multivariate Normal Distribution*. Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, Nr. 1, pp. 197-206, 1956.

[SW81a] T. Smith, M. S. Waterman. *Comparison of biosequences*. Advances in Applied Mathematics, Vol. 2, pp. 482-489, 1981.

[SW81b] T. Smith, M. S. Waterman. *Identification of common molecular subsequences*. Journal of Molecular Biology 147:195-197, 1981.

[Tho99] J. D. Thompson, F. Plewniak, O. Poch. *A comprehensive comparison of multiple sequence alignment programs*. Nucleic Acids Research, Vol. 27, Nr. 13, pp. 2682-2690, 1999. `http://nar.oupjournals.org/cgi/reprint/27/13/2682.pdf`

[Tsu02] K. Tsuda, T. Kin, K. Asai. *Marginalized kernels for biological sequences*. Bioinformatics 18 (Suppl 1), pp. 268-275, 2002.

[TT78] R. A. Tapia and J. R. Thompson. *Nonparametric Density Estimation*. John Hopkins University Press, Maryland, 1978.

[Tur] B. A. Turlach. *Bandwidth selection in kernel density estimation: a rewiew*. Statistik und Ökonometrie 9307, Humboldt Universität Berlin, undatiert.

[Vap95] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, Berlin, 1995.

[Vap98] V. Vapnik. *Statistical Learning Theory*. John Wiley, 1998.

[Wat84] M. S. Waterman. *Efficient Sequence Alignment Algorithms*. Journal of Theoretical Biology, Vol. 108, pp. 333-337, 1984. `http://www-hto.usc.edu/papers/msw_papers/msw-053.pdf`

[WL83] W. J. Wilbur, D. J. Lipman. *Rapid similarity search of nucleic acid and protein data banks*. Proceedings of National Academy Science of the USA Vol. 80, pp. 726-730, 1983.

[WM03] S. Weiss, H. M. Müller. *The contribution of EEG coherence to the investigation of language*. Brain & Language, Vol. 85, pp. 325-343, 2003.

[WR96] S. Weiss, P. Rappelsberger. *EEG coherence within the 13-18 Hz band as a correlate of a distinct lexical organization of concrete and abstract nouns in humans*. Neuroscience Letters, 209, pp. 17-20, 1996.

[Zsch95] S. Zschocke. *Klinische Elektroenzephalographie*. Springer Verlag, Berlin, 1995.