

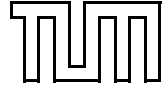
Technische Universität München

Fakultät für Informatik

Masterarbeit in Informatik

Teilüberwachtes Lernen mit Untergraphen

Kamil Ciosek



Technische Universität München

Fakultät für Informatik

Masterarbeit in Informatik

Teilüberwachtes Lernen mit Untergraphen

Semi-supervised Learning from Subgraphs

Aufgabensteller:	Prof. Dr. Stefan Kramer
Betreuer:	Dipl.-Bioinf. Tobias Girschick
Bearbeiter:	Kamil Ciosek
Abgabedatum:	14. September 2010

Ich versichere, dass ich diese Masterarbeit selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

14. September 2010

Zusammenfassung

In silico-Verfahren spielen eine zunehmend wichtige Rolle in der Entwicklung neuer Medikamente, da sie billiger als Laborexperimente sind. Das Ziel dieser Arbeit ist es, Möglichkeiten zu untersuchen, wie die Effizienz der Klassifizierungs- und Regressionswerkzeugen auf Molekülsätzen verbessert werden kann. Es wird der Versuch unternommen, sowohl Merkmalsuche als auch das Lernverfahren teilüberwacht auszurichten, also auch das Potenzial der unetikettierten Moleküle zu verwenden. Dies hat breite praktische Anwendung, da unetikettierte Molekülgraphen in großer Vielfalt frei verfügbar sind. Die Moleküle werden als Graphen behandelt, als Deskriptoren werden ihre Untergraphen verwendet. Zuerst werden mehrere Ansätze zur Bestimmung der optimalen Deskriptorenmenge untersucht und verglichen. Anschließend folgt eine Besprechung der Generalisierungsverfahren, wobei sowohl klassische Algorithmen wie kNN, (T)SVM und Entscheidungsbäume als auch neue Ideen aus der Forschung zur teilüberwachten Regression berücksichtigt werden. Der Leitgedanke bleibt stets, Erkenntnisse aus anderen Zweigen des maschinellen Lernens auf das Lernen mit Graphen zu übertragen. Es kommen auch solche Verfahren zum Einsatz, die im Kontext der molekulären Graphen noch nicht verwendet wurden. Der Fokus der Arbeit liegt auf Merkmalsuche und dem Lernverfahren; Untergraphenextraktion wird nur am Rande angesprochen. Am Ende werden mehrere Varianten der Pipeline ausgewertet. Es wird festgestellt, dass das Vorhaben, unetikettierte Daten miteinzubeziehen, nicht geglückt ist.

Abstract

In silico-methods are increasingly important in drug development because they are cheaper than lab experiments. The purpose of this thesis is to examine possibilities of improving the performance of classification and regression tools applied to molecular datasets. It is attempted to use the semi-supervised paradigm for both feature extraction and generalisation so as to uncover the potential of unlabeled molecules. This has wide practical applications since unlabeled molecular graphs of wide variety are freely available. Molecules are treated as graphs and their subgraphs are used for descriptors. First, diverse ways of finding the optimal set of descriptors are analysed and compared. Next, diverse learners are described, including both classic algorithms like kNN, (T)SVM and decision trees and new developments in semi-supervised regression. The core idea always remains to carry results from other areas of machine learning over to learning from graphs. Some of the used algorithms have not been applied to molecular graphs before. The focus is on feature search and the learner, subgraph extraction is not treated in any great depth. Finally, the performance of several variants of the pipeline is evaluated. It is concluded that the attempt to improve learner quality by incorporating unlabeled data has been unsuccessful.

Inhalt

Inhalt	1
Einführung	5
Definitionen	7
1 Merkmalsuche und Untergraphenextraktion	9
1.1 Merkmalsuche für Klassifizierungsprobleme	12
1.1.1 Die Baseline-Ansätze – χ^2 und Supportschränke	12
1.1.2 Nach einer Score-Funktion gerichtete Suche	12
1.1.3 Skalarprodukt-Score	14
1.1.4 Submodulare Inkonsistenzenfunktion	15
1.1.5 Wrappers	16
1.1.6 Markovsche Decke	17
1.1.7 Vergleich der Ansätze	19
1.2 Verfahren für Regression	20
1.2.1 Pearson-Korrelation	20
1.2.2 Mutual Information	21
1.2.3 Rayleigh-Koeffizient	21
1.2.4 Hauptkomponentenanalyse	24
1.2.5 Vergleich der Ansätze	24
1.3 Ausblick: Konstruktion von Merkmalen	25
1.4 Anmerkungen zur Graphenextraktion	25
2 Klassifizierungs- und Regressionsverfahren	29
2.1 Nächster Nachbar	29
2.2 Entscheidungsbäume und J48graft	30
2.3 SVMs, TSVMs und Varianten davon	31
2.3.1 Überwachtes Lernen	31
2.3.2 Teilüberwachtes Lernen	34
2.3.3 Regression	36
2.3.4 Die Unmöglichkeit transduktiver Regression mit SVMs	37
2.4 Transduktive Regression	37
2.4.1 Zweischrittverfahren	37
2.4.2 Direkte Regularisierung	38
2.4.3 Regularisierung in einer Funktionsfamilie	39
2.4.4 Metrikbasierte Ansätze	40
2.4.5 Vergleich zwischen den Ansätzen	41

3	Pipeline und Auswertung	43
3.1	Methoden der Auswertung	43
3.1.1	Die Heuristiken im überwachten Fall	43
3.1.2	Gefahren der statistischen Auswertung	44
3.1.3	Der ROC-Raum	45
3.1.4	Der teilüberwachte Fall	46
3.1.5	Auswertungsgrößen bei Regression	47
3.2	Das benutzte Auswertungsverfahren	47
3.3	Die benutzten Datensätze	49
3.4	Die Pipeline	50
3.5	Interpretation der Ergebnisse - Klassifizierung	51
3.6	Interpretation der Ergebnisse - Regression	56
4	Schlussfolgerungen und Weiterentwicklung	63
	Literatur	65
	Anhang A Ergebnisse - Klassifizierung	71
	Anhang B Ergebnisse - Regression	75

Notation

\mathbb{B}	Boolsche Werte
kalligraphierte Buchstaben (z.B. $\mathcal{G}, \mathcal{A}, \dots$)	Mengen von Merkmalen oder Graphen
fette Buchstaben (z. B. $\mathbf{g}_i, \mathbf{s}_i, \dots$)	Merkmale, Graphen, logische Prädikate, abstrakte Trainingsobjekte, Knoten
normale Buchstaben (z. B. o_n, e, \dots)	Zahlen
$ \cdot $	Kardinalität einer Menge
V	Menge der Werte der Zielvariable
T	Die Zielvariable
zweibuchstabige Symbole (z.B. $MV^{\mathbb{B}}, GV^{\mathbb{A}}$)	Vektoren (Boolsch mit Elementen anderer Mengen)
$el(MV^{\mathbb{B}}, i)$	Das i -te Element des Vektors $MV^{\mathbb{B}}$
$MV^{\mathbb{B}} _{\mathcal{L}}$	Verkürzung des Vektors $MV^{\mathbb{B}}$ auf Objekte aus der Menge L
$support_{\mathcal{G}}(\mathbf{s})$	Anzahl der Graphen in \mathcal{G} , wo \mathbf{s} ein Untergraph ist
\subseteq	Untermengen- oder Untergraphenrelation, je nach Kontext
$I(\mathcal{A}, \mathcal{B} \mathcal{C})$	bedingte Unabhängigkeit
TP, FP	wahre, falsche Positive (siehe Abschnitt 3.1.3)
d	Anzahl der Attribute
n	Anzahl der betrachteten Menge der Graphen
l	Anzahl der der etikettierten Graphen

Einführung

Das Lernen mit Graphen ist ein praxisnahes Problem, das unter anderem in der *in silico*-Medikamentenforschung oft anzutreffen ist und dessen gute Lösung dort erhebliche Ersparnisse möglich macht. Formal betrachtet ist das Ziel eines solchen Klassifizierungs- oder Regressionsverfahrens, einen Algorithmus zu konstruieren, der den Molekülgraphen als Eingabe einliest und auf dessen Basis den Wert einer Zielvariable möglichst exakt prognostiziert. Die Zielvariable kann dabei sowohl ein kategorischer Wert sein als auch eine rationale Zahl, die zum Beispiel für die Toxizität des betrachteten Stoffes in einem Organismus steht.

In dieser Arbeit wird die Grundannahme verfolgt, dass die Zielvariable tatsächlich von der topologischen Struktur des Moleküls determiniert ist und nicht zum Beispiel der Einbettung des Moleküls in den dreidimensionalen Raum: Kurzum, wir nehmen an, dass die Graphenstruktur eine gute Näherung des Moleküls ist. Dies bedeutet, dass man die Menge der Untergraphen des Graphen, der die chemische Struktur des Moleküls modelliert als Eingabe für den Algorithmus benutzen kann. Es gibt auch andere Ansätze, Moleküle zu beschreiben, wie zum Beispiel durch Zahlen, die chemischen Eigenschaften entsprechen (etwa die Molekülmasse, es sind aber auch viel kompliziertere Ansätze möglich [85]) oder durch die 3D-Struktur des Moleküls [81], sie werden hier nicht untersucht.

Es ist ferner in dem Bereich der Bioinformatik oft der Fall, dass die Kosten, Moleküle auf bestimmte Eigenschaften zu überprüfen, hoch ausfallen, da dies Experimente erfordert, die von Menschen durchgeführt werden müssen. Die zu erforschenden chemischen Strukturen selber sind aber bekannt und können problemlos in einem Computer modelliert werden. Deswegen erscheint es zweckmäßig, Techniken des teilüberwachten Lernens auf Graphen anzuwenden. Beim teilüberwachten Lernen handelt es sich um eine Situation, in der man neben einem Trainingssatz von Objekten, deren Labels bekannt sind auch über eine zusätzliche Menge verfügt, wo die Objekte bekannt sind, aber nicht ihr Etikett. Das Ziel ist es, eine Regel für unbekannte Objekte zu generieren, wobei man die Annahme macht, dass das zusätzliche Wissen, dass man durch die etikettenlosen Objekte über die Verteilung bekommt, nützlich ist. Eine verwandte Aufgabe ist die der Transduktion: Der Unterschied besteht darin, dass man im Falle der Transduktion keine Entscheidungsregel mehr generieren möchte, sondern lediglich die Etiketten für die im voraus bekannte Menge der etikettenlosen Objekte. Man kann dies als ein einfacheres Problem betrachten [88].

Die Generalisierungsverfahren werden üblicherweise in drei konzeptuelle Phasen aufgeteilt: Zuerst werden Untergraphen extrahiert, dann wird eine Menge von „interessanten“ Untergraphen selektiert und am Ende wird der eigentliche Schritt der Klassifizierung oder Regression durchgeführt. Diese Arbeit handelt hauptsächlich von der zweiten und dritten Phase. Im Kapitel 1 werden mehrere Ansätze zur Merkmalsuche vorgestellt, die dann miteinander verglichen werden. Im Kapitel 2 werden die klassischen Verfahren (kNN, Entscheidungsbäume und SVM) sowie neuere Ansätze für teilüberwachte Regression besprochen. Die Graphenextraktion selber wird nur kurz angesprochen (Abschnitt 1.4). Im Kapitel 3 wird die Methodologie der Auswertung besprochen und die Ergebnisse interpretiert. Tabellen mit vollständigen Ergebnissen befinden sich in Anhängen A und B.

Definitionen

Wir werden jetzt einige grundlegende Begriffe formell definieren, die in den nachfolgenden Kapiteln immer wieder zur Anwendung kommen. Sie entsprechen der Ausgabe des Extraktionsverfahrens.

Definition 1. Die ETIKETTFUNKTION l ist eine stochastische Relation $l : \mathcal{G} \rightarrow V$, wobei V die Menge der möglichen Werte der Zielvariable ist.¹

Definition 2. Ein MERKMALVEKTOR für einen Graphen $\mathbf{g} \in \mathcal{G}$ ist ein Vektor $MV(\mathbf{g})$, so dass folgendes gilt.

1. $el(MV^{\mathbb{B}}(\mathbf{g}), i) \Leftrightarrow \mathbf{s}_i \subseteq \mathbf{g}$
2. $el(MV^{\{x,y\}}(\mathbf{g}), i) = x$ wenn $\mathbf{s}_i \subseteq \mathbf{g}$ ansonsten y ; $x, y \in \mathbb{R}$,

Definition 3. Ein GRAPHENVEKTOR für ein Merkmal $\mathbf{s} \in \mathcal{S}$ ist ein Vektor $GV(\mathbf{s})$, so dass folgendes gilt.

1. $el(GV^{\mathbb{B}}(\mathbf{s}), i) \Leftrightarrow \mathbf{s} \subseteq \mathbf{g}_i$
2. $el(GV^{\{x,y\}}(\mathbf{s}), i) = x$ wenn $\mathbf{s} \subseteq \mathbf{g}_i$ ansonsten y ; $x, y \in \mathbb{R}$,

Die Begriffe der Merkmalvektoren und Graphenvektoren sind natürlich völlig redundant, sie entsprechen zweier Ansichten gleicher Daten, die in einer Implementierung nur einmal gespeichert werden. Sie werden hier eingeführt, da ansonsten die Notationen in den nachfolgenden Kapiteln kompliziert werden.

Definition 4. Der ZIELVEKTOR für ist ein Vektor $TV = [y_1, \dots, y_n]^T$ mit Werten aus V , so dass $y_i = l(G_i)$. Wenn l verschiedene Ausgaben für die gleiche Eingabe zulässt, ist er als Zufallsvektor zu interpretieren.

Wenn wir teilüberwachtes Lernen betrachten, so ist die Anzahl der Graphenvektoren größer als die Länge des Zielvektors.

Die Aufgabe des Klassifizierungsverfahrens kann man also so zusammenfassen: Wenn TV und $\{MV(\mathbf{g}_i)\}$ (oder $\{GV(\mathbf{s}_i)\}$) gegeben sind, gilt es, eine Möglichst gute Annäherung von l zu berechnen.

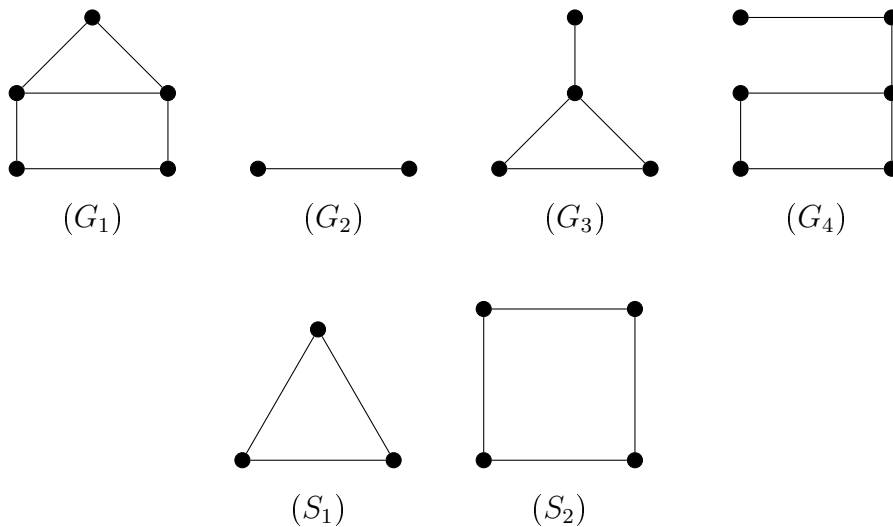
¹In der Literatur wird l oft als eine Funktion betrachtet (in unserem Fall wäre das eine Funktion $\mathbb{B}^n \rightarrow \mathbb{V}$). Eine solche Definition zieht nach sich, dass die Eingabemenge keine Inkonsistenzen drin hat, was zwar generell sinnvoll ist, manchmal aber wegen verrauschter Daten oder unrichtigen Annahmen nicht der Wirklichkeit entspricht. Die Aufgabe des Lernens kann dann als Versuch, die Gesamtdefinition der Funktion anhand fragmentarischer Informationen herzustellen, gedeutet werden [47].

Kapitel 1

Merkmalsuche und Untergraphenextraktion

In diesem Kapitel wird die Frage behandelt, welche von den extrahierten Untergraphen fürs Lernen am meisten geeignet sind. Bei Graphenextraktion muss Folgendes bedacht werden: Die Anzahl der Untergraphen eines bestimmten Graphen ist exponentiell in seiner Größe, was nur teilweise dadurch gemindert wird, dass man nur nach Untergraphen sucht, die in allen in dem betrachteten Datensatz vorkommenden Molekülen zu finden sind. Außerdem tun sich die meisten Klassifizierer damit schwer, mit sehr großen Mengen von Merkmalen zu arbeiten. Daher ist es nötig, die zu betrachtende Menge von Untergraphen zu beschränken. Da man dabei die Fähigkeit verliert, einige topologische Eigenschaften der Graphen voneinander zu unterscheiden, ist es nötig, dass man genau überlegt, welche Merkmale zu eliminieren sind, so dass die Fähigkeit des Klassifizierers, genaue Voraussagen zu treffen, möglichst wenig beeinträchtigt wird. Ein weiterer Grund, warum es nötig ist, Merkmalsuche durchzuführen ist, dass manche Klassifizierer viel schlechtere Lerneigenschaften haben, wenn sie irrelevante Merkmale als Eingabe bekommen. Dies ist zum Beispiel bei kNN der Fall — eine formelle Analyse zeigt [51], dass die zur Erreichung einer bestimmten Trefferquote notwendigen Trainingsobjekte unter bestimmten Annahmen schnell wächst, wenn die Anzahl der irrelevanten Attribute größer wird. Andere Klassifizierer, wie zum Beispiel naiver Bayes, sind dafür weniger anfällig, tendieren aber dazu, schlechter abzuschneiden, wenn in den Daten viele korrelierten Merkmale sind, selbst wenn diese bezüglich der Zielvariable relevant sind [47]. Im Übrigen sind die Ergebnisse der Lernalgorithmen bei kleineren Merkmalmengen einfacher zu interpretieren, bei automatischer Regelgenerierung können sie weniger Überanpassung aufweisen. Ferner kann die Existenz einer kleinen Menge von Attributen ohne Inkonsistenzen bezüglich der Zielvariable als Begründung dafür dienen, dass der Datensatz eine interessante Struktur hat (es kann ein Zusammenhang zwischen der Größe eines Datensatzes und der Existenz zufälliger Merkmalmengen ohne Inkonsistenzen festgestellt werden, die eine bestimmte Anzahl der Merkmale haben — weniger Merkmale deutet auf eine Besonderheit der Struktur des Datensatzes hin [14]). Im gleichen Zuge könnte man sagen, dass der unüberwachte Teil der Merkmalsuche grundsätzlich die Struktur der Eingabe zusammenfasst, also in gewissem Sinne eine Form der Datenkomprimierung ist [37]. Es kann auch nach dem in Machine Learning oft angewandten Prinzip des Ockhamschen Rasiermessers argumentiert werden, dass eine kleinere Anzahl der Merkmale zu besserer Generalisierungsfähigkeit führt [14], da sie die Komplexität und Anzahl der betrachteten Hypothesen reduziert. Dabei ist zu beachten, dass dieses Prinzip kein stichfestes Faktum von universaler Gültigkeit darstellt, sondern nur ein Volkstheorem ist (obgleich bei bestimmten Verfahren nachgewiesen werden kann, dass es tatsächlich optimal ist — dies kann aber nur unter Zugabe zusätzlicher Annahmen erfolgen). Wenn man immer nach diesem Prinzip verfahren würde, so würde man

Abbildung 1 Das XOR-Problem für Graphen. $L(\mathbf{g}_1) = L(\mathbf{g}_2) = 0$, $L(\mathbf{g}_3) = L(\mathbf{g}_4) = 1$.



zum Beispiel in einer Datenbank, die Angaben zu Personen und deren Eigenschaften hat, die Ausweisnummer als das einzig relevante Merkmal auswählen, obwohl dieses Merkmal denkbar schlechtes Verallgemeinerungspotenzial aufweist.

Von einem idealisierten Standpunkt aus betrachtet wäre es der Zweck der Merkmalsuche die Untermenge der Merkmale zu identifizieren, die für die Berechnung der Zielvariable entscheidend ist. Da die Relevanz einer Merkmalmenge aber nicht auf eine eindeutige Art und Weise zu definieren ist, behilft man sich mit Ansätzen, die dieses Ziel auf jene oder andere Art und Weise approximieren. So kann man eine heuristisch motivierte Potenzialfunktion nehmen und Merkmalsuche auf kombinatorische Optimierung reduzieren; man kann auch versuchen, die Wahrscheinlichkeitsverteilung der Zielvariable und der Merkmale — die wir ja nicht kennen — zu modellieren oder es sich zum Ziel zu setzen bezüglich eines gewählten Klassifizierers die Trefferquote zu steigern [27]. Trotz der Vielzahl der möglichen Vorgehensweisen sind bei der Merkmalsuche stets zwei allgemeine Postulate zu beachten, die von verschiedenen Ansätzen auf verschiedene Art und Weise zur Geltung gebracht werden. Erstens sollen die ausgewählten Merkmale mit den Etiketten korreliert sein, damit der Klassifizierer genug Informationen für genaue Prädiktion hat. Zweitens sollen sie untereinander möglichst schwach korreliert sein, damit keine Redundanz entsteht [68]. Diese Postulate können nach dem Wissen des Autors nicht in einer allgemeinen, für alle Ansätze richtigen Relevanzformel niedergeschrieben werden, da z. B. alleine die Auswahl eines Korrelationsmaßes als Modellierungsannahme zu werten ist. Außerdem können die Korrelationen zwischen den Merkmalen eine stark nichtlineare Form haben oder es kann sein, dass einige Merkmale zwar alleine betrachtet überhaupt nicht aussagekräftig sind, zusammen aber sehr genaue Prädiktion ermöglichen [37]. Das wohl bekannteste Beispiel dafür ist das XOR-Problem (siehe Abbildung 1). Es liegt also nahe zu behaupten, dass die Auswahl der Art und Weise, wie bei der Merkmalsuche die Korrelation verschiedener Merkmale untereinander und mit der Zielvariable miteinbezogen wird, eng mit dem induktiven Bias bei Klassifizierung oder Regression verbunden ist, da man jeweils über die Struktur des Raumes der erlaubten Strukturen Annahmen macht. Eine Möglichkeit, diesen Zusammenhang zu interpretieren, führt zu der Annahme, dass die Optimalität der Merkmalsuche stets bezüglich eines Klassifizierers definiert werden sollte [47]. Dies führt zu Wrapper-Ansätzen, die wir später behandeln werden.

Um die Probleme, die man bei der Definition der Relevanz von Merkmalmengen hat, zu veranschaulichen, betrachten wir jetzt zwei Beispiele. Sie stammen von Kohavi und John [47]. Zuerst werden wir zeigen, dass die Relevanz einer Merkmalmenge nicht zwangsläufig das optimale Lernverhalten nach sich zieht. Nehmen wir an, dass wir drei Merkmale, \mathbf{x}_1 , \mathbf{x}_2 und \mathbf{x}_3 haben, dass die Verteilung der Objekte bezüglich der drei Merkmale gleichmäßig ist und dass sich die Zielvariable aus der Formel $\mathbf{x}_1 \wedge \mathbf{x}_2 \vee \mathbf{x}_3$ ergibt. Betrachten wir jetzt einen Klassifizierer, der mit Hypothesen arbeitet, die Konjunktionen von Merkmalen sind. Er erreicht dann das optimale Verhalten, wenn lediglich \mathbf{x}_3 in die betrachtete Merkmalmenge kommt — mehr Variablen verschlechtern seine Trefferquote, und das obwohl alle drei Merkmale jede sinnvolle Definition der Relevanz bezüglich der Zielvariable offensichtlich erfüllen. Zweitens impliziert — was vielleicht erstaunlicher ist — das optimale Verhalten eines Klassifizierers nicht, dass die Merkmalmenge relevant ist. Betrachten wir zum Beispiel ein eingeschränktes Perzeptron mit festem Schwellenwert null. Nun stelle man sich vor, dass dieses eine Eingabe hat, die immer auf eins gesetzt wird. Ein Merkmal, das immer den gleichen Wert hat, kann nicht relevant sein und doch erhöht es die Klassifizierungsfähigkeit des Perzeptrons auf die eines mit variablem Schwellenwert.

Eine erwünschte Eigenschaft der Merkmalsuche ist die Einsetzbarkeit beim unüberwachten Lernen. Intuitiv gesehen können die unetikettierten Strukturen dazu verwendet werden, die Korrelation zwischen den Merkmalen besser einzuschätzen, aber nicht zwischen den Merkmalen und der Zielvariable. Ebenfalls wünschenswert ist es, dass sich die Merkmalauswahl in den Prozess der Extraktion der Untergraphen einbinden lässt [68]. Dies macht es möglich, mit viel größeren Datensätzen zu arbeiten, beziehungsweise den minimalen Support der erfassten Untergraphen zu reduzieren, da bei der Merkmalsuche ganze Suchzweige nicht betreten werden müssen und das Ganze infolgedessen viel schneller wird. Es kann sein, dass die Möglichkeit, mehr Untergraphen zu betrachten auch die Trefferquote des ganzen Verfahrens steigert. Nichtsdestotrotz kann man bei den theoretischen Überlegungen immer davon ausgehen, dass die Merkmalsuche nach abgeschlossener Extraktion als ein Filterungsschritt stattfindet. Wie das tatsächlich implementiert wird, ist eine andere Sache.

Die Definition des Extraktionsverfahrens setzt nicht voraus, dass alle Untergraphen gefunden werden. Angesichts der vorigen Überlegungen ist es aber erstrebenswert, dass möglichst wenig brauchbare Merkmale bereits wegen der Extraktion eliminiert werden. Es ist üblich, für die extrahierten Untergraphen eine Support-Schranke zu setzen.

Definition 5. Ein MERKMALSUCHVERFAHREN ist ein Algorithmus, der $\subseteq: \mathcal{S} \times \mathcal{G}$ oder $\subseteq: \mathcal{S} \times \mathcal{S} \cup \mathcal{G}$ und l entgegennimmt und $\mathcal{S}' \subseteq \mathcal{S}$ zurückliefert.

Der Zweck der Merkmalsuche ist es natürlich, dass die ausgewählte Menge \mathcal{S}' im Hinblick auf die Zielvariable möglichst repräsentativ für das ganze \mathcal{S} ist, es ist aber, wie oben beschrieben, schwierig, diese Bedingung zu formalisieren und dabei allgemein zu bleiben.

Implementationstechnisch gesehen werden zur Repräsentation von \subseteq oft Boolesche oder reelle Vektoren verwendet. Im letzteren Fall kann die Abwesenheit eines Graphen üblicherweise entweder als -1 oder als 0 markiert werden.

Man kann jetzt gleich zwei Begriffe einführen, die bei der Graphenextraktion nützlich sind.

Definition 6. Die Menge von Untergraphen \mathcal{S} ist bezüglich der Datenbank \mathcal{G} genau dann GESCHLOSSEN, wenn $\forall \mathbf{s}_i \subseteq \mathbf{s}_j. \text{support}_{\mathcal{G}}(\mathbf{s}_i) > \text{support}_{\mathcal{G}}(\mathbf{s}_j)$.

Definition 7. Ein Merkmal \mathbf{s} ist genau dann REDUNDANT, wenn es ein anderes Merkmal \mathbf{s}' gibt, so dass $\forall \mathbf{g} \in \mathcal{G}. \mathbf{s} \subseteq \mathbf{g} \Leftrightarrow \mathbf{s}' \subseteq \mathbf{g}$.

Es kann sehr einfach gezeigt werden, dass redundante Merkmale beim Lernen nicht helfen (das heißt man kann nur einen davon belassen). Es ist ebenfalls sofort ersichtlich, dass man bei Extraktion von Untergraphen, die nicht geschlossen sind, redundante Merkmale einführt.

In den nachfolgenden Abschnitten werden einige Ansätze für die Merkmalsuche vorgestellt. Sie werden mithilfe von sechs Kriterien verglichen: Es wird untersucht, inwiefern die Korrelation von Merkmalen jeweils untereinander und mit der Zielvariable berücksichtigt wird, ob die Gitterstruktur unter den Untergraphen (das heißt die zusätzliche Information über die Relation \subseteq) für die Anwendung des Verfahrens notwendig ist, ob sich das Verfahren auch für teilüberwachtes Lernen eignet (inwiefern die nicht etikettierten Graphen zur Geltung kommen), und kategorischen Variablen anwendbar ist, sowie ob er sich in die Extraktion der Untergraphen einbinden lässt. Bei Ansätzen, die für Klassifizierungsprobleme entwickelt wurden, wird auch die eventuelle Übertragung auf Regression besprochen.

1.1 Merkmalsuche für Klassifizierungsprobleme

1.1.1 Die Baseline-Ansätze – χ^2 und Supportschranke

Einer der einfachsten Ansätze zur Auswahl von Variablen liefern statistische Signifikanzmaße. Ein häufig verwendetes Beispiel davon ist das χ^2 -Maß. Es wird wie als $\chi^2 = \sum_c \frac{(o_c - e_c)^2}{e_c}$ definiert [29]. Das Symbol o_c entspricht der Anzahl der Graphen mit dem gerade betrachteten Merkmal, die das Etikett c haben, e_c dem erwarteten Wert dieser Zahl, der dann zu erwarten wäre, wenn es zwischen dem Vorhandensein des Untergraphen und dem Etikett keine Korrelation gäbe, und der als $c_c n^{-1} s$ definiert ist. c_c steht dabei für die Anzahl der Graphen mit dem Etikett c , s ist die Anzahl der Graphen mit dem betrachteten Untergraphen und n ist die Zahl der Graphen überhaupt. Dieser Wert gilt für Klassifizierungsprobleme (auch mit mehreren Kategorien), eine direkte Übertragung auf Regression ist jedoch nicht möglich. Die Gitterstruktur der Menge der Untergraphen wird nicht benötigt. Es wird hier nur die Korrelation der Merkmale mit der Zielfunktion berücksichtigt, nicht aber untereinander. Teilüberwachte Filterung ist nicht möglich, da für die Berechnung des χ^2 -Maßes das Wissen über die Etiketten erforderlich ist.

Bei binärer Klassifizierung ($V = \{0, 1\}$) lässt sich die Berechnung des Maßes in die Extraktion einbauen, was diese bei der Einführung einer Schranke schneller macht, da das χ^2 -Maß konvex ist und somit dank der folgenden Ungleichheit Pruning ermöglicht [60]. Man kann bei $V = \{0, 1\}$ sagen, dass das Maß nur von den Werten s , und o_1 abhängt: $\chi^2 = \chi^2(s, o_1)$ da $o_0 = s - o_1$ und die Werte von n , c_c für alle Untergraphen gleich sind.

$$\mathbf{s}_1, \dots, \mathbf{s}_n \subseteq \mathbf{s} \Rightarrow \chi^2(s(\mathbf{s}), o_1(\mathbf{s})) \leq \min_i \max\{\chi^2(o_1(\mathbf{s}_i), o_1(\mathbf{s}_i)), \chi^2(s(\mathbf{s}_i) - o_1(\mathbf{s}_i))\}$$

Ein anderer einfacher Ansatz ist die Supportschranke - man übernimmt nur die Merkmale, die oft genug vorkommen und kann dann anschließend einen anderen Filterungsmechanismus anwenden. Der Support wird als eine Zahl angegeben, die mit der Anzahl der Graphen mit dem jeweiligen Untergraphen verglichen wird, wobei man sowohl die unetikettierten als auch die etikettierten Untergraphen berücksichtigen kann.

1.1.2 Nach einer Score-Funktion gerichtete Suche

Einer der am weitesten verbreiteten Ansätze für Merkmalsuche ist es, zur Beurteilung von Untermengen von \mathcal{S} eine Score-Funktion zu konstruieren und dann einen Teil der Potenzmenge zu durchsuchen [10]. Dann sucht man in einer gewissen Zahl der Iterationen, die ein Parameter des Algorithmus ist. Das ganze wird bis zum Erreichen einer festgelegten Anzahl an Merkmalen wiederholt, wobei die Score-Funktion in den nachfolgenden Schritten die bereits errechnete Menge berücksichtigt. Die Suche kann nun verschieden durchgeführt werden. Es sind grundsätzlich zwei Dinge festzulegen: die Art und Weise, wie man für einen bestimmten Untergraphen die Menge seiner Nachbarn bestimmt und wie einer davon ausgewählt wird

[27]. Man wählt natürlich immer aus der Menge $\mathcal{C} = \mathcal{S} \setminus \mathcal{S}'$, da die bereits ausgewählten Merkmale nicht ein zweites Mal hinzugefügt werden können. Eine gute Möglichkeit, nach Nachbarn zu suchen, ist die Relation \subseteq zu nutzen und in einem Schritt die Graphen zu betrachten, die Übergraphen des gerade betrachteten Graphen \mathbf{m} sind (das heißt die Menge $\{\mathbf{m}' \in \mathcal{C} : \mathbf{m} \subseteq \mathbf{m}'\}$). Wenn die Information zum Untergraphenisomorphismus unter den Merkmalen nicht zur Verfügung steht, so kann man den Abstand auch anders definieren, zum Beispiel als Hamming-Distanz. Dann nimmt man einfach man nimmt dann in jedem Schritt eine festgelegte Zahl der Graphen mit dem geringsten Abstand zu \mathbf{m} . Eine dritte, und wohl die einfachste Möglichkeit ist es, einfach die ganze Menge \mathcal{C} zu nehmen. Die Zweite Frage ist es, wie der Nachbar errechnet wird. Das wohl einfachste ist es, den Nachbarn mit dem jeweils höchstem Wert der Score-Funktion zu nehmen, also den Greedy-Ansatz anzuwenden. Man kann auch stochastisch vorgehen und den Nachbarn so zufällig auswählen, dass diejenigen mit größerem Score-Wert auch mit größerer Wahrscheinlichkeit ausgewählt werden. Es ist auch möglich, mit einer bestimmten Wahrscheinlichkeit einen zufälligen Nachbarn auszuwählen (siehe Algorithmus 1). Der gesamte Algorithmus wird hier nicht angegeben, er entspricht einer klassischen linearen Suche: man sucht sich in jedem Schritt den nächsten Nachbarn aus und wählt am Ende denjenigen von den besuchten, wo Score am höchsten war.

Die Vorteile dieses Algorithmus sind seine Einbindbarkeit in die Extraktion der Untergraphen, Eignung für teilüberwachte Filterung sowie dass er je nach Score-Funktion sowohl die Korrelation mit der Zielvariable als auch der Merkmale untereinander berücksichtigt. Es sei hier bemerkt, die Suche in der hier vorgestellten Variante auf dem Vorwärts-Ansatz beruht. Man kann sich aber auch leicht modifizierte Varianten davon vorstellen, wo die Suche rückwärts erfolgt (man startet mit der vollen Merkmalmenge und eliminiert die Merkmale der Reihe nach, bis eine bestimmte Anzahl davon erreicht wird), oder dass eine kompliziertere Metaheuristik für die Suche angewandt wird - zum Beispiel best-first, genetische Suche, oder man könnte neben dem Einfügen eines Merkmals in die betrachtete Menge zusätzliche Operatoren definieren [47] (zum Beispiel das Entfernen eines Merkmals, oder solche, die mit mehreren Merkmalen zeitgleich arbeiten). Ein Argument gegen die Greedy-Suche ist es, dass man relativ einfach künstliche Datensätze erfinden kann, bei denen sie fehlschlägt. Betrachten wir nun (das Beispiel kommt aus [47]) einen Datensatz mit den Merkmalen $\mathbf{a}_0, \mathbf{a}_1, \mathbf{b}_0, \mathbf{b}_1, \mathbf{i}, \mathbf{c}$,

Algorithmus 1 Möglichkeiten für die Auswahl der Nachbarn.

WähleNachbarn-Stochastisch(*merkmal*)

begin

kandidaten = NachbarnMenge(*merkmal*, \mathcal{S}')

scores = map ($\lambda \mathbf{m}. \text{Score } \textit{merkmal } S$) *kandidaten*

 Normalisiere(*scores*); //macht alle Scores positiv durch Addieren einer Konstante

randomNumber = makeRandom(0, $\sum_i \textit{scores}_i$) //z.B. gleichmäßige Verteilung

 return(*kandidaten*_{*w*}, wobei

$$\sum_{i < w} \textit{scores}_i < \textit{randomNumber} \wedge \sum_{i \leq w} \textit{scores}_i \geq \textit{randomNumber}$$

end

WähleNachbarn-Greedy(*merkmal*, \mathcal{S}')

begin

kandidaten = NachbarnMenge(*merkmal*, \mathcal{S}')

 return($s \in \mathcal{S} \setminus \mathcal{S}'$ mit dem größten Score(c, \mathcal{S}'))

end

wo sich die Zielvariable aus der Formel $\mathbf{a}_0 \wedge \mathbf{a}_1 \vee \mathbf{b}_0 \wedge \mathbf{b}_1$ ergibt, das Merkmal i zufällige Werte hat und das Merkmal c bei durchschnittlich 75% der Objekte mit der Zielvariable korrespondiert. Dann wählen die Greedy-Ansätze zuerst das Merkmal c , obwohl der eigentlich nichts bringt, was sich dann auf die Beurteilung der anderen Merkmale je nach Score-Funktion negativ auswirken kann. Zu der Häufigkeit solcher Situationen in echten Datensätzen machen wir jetzt keine Aussage. Der gesamte Algorithmus der Merkmalsuche kann also als eine Zusammensetzung der Suchmethode und einer Score-Funktion verstanden werden. Zu der Score-Funktion kann man sagen, dass sie grundsätzlich immer an die Zielvariable und an den Klassifizierer angepasst werden muss. So könnte man zum Beispiel behaupten, alle Score-Funktionen sollen monoton sein, da dieses (unter der Annahme, dass die Daten nicht verrauscht sind und aus einer gemeinsamen Wahrscheinlichkeitsverteilung stemmen) für den optimalen Bayes-Klassifizierer sinnvoll ist [47] (und nebenher effizientes Pruning ermöglicht). Wie aber bereits in der Einführung besprochen wurde, muss dies nicht für andere Klassifizierer gelten, die z. B. für wenig relevante Attribute anfällig sind. Es sind auch kompliziertere Varianten von Suchverfahren möglich [47], [10]. Der Hintergrund von solchen Ansätzen ist es, den größten Nachteil der Greedy-Suche (den eingeschränkten Suchraum unter Untermengen) zu minimieren, indem man diesen Raum anreichert — die ganze Potenzmenge kann man nicht durchsuchen, es sei denn, das Evaluationskriterium wäre so konstruiert, dass besonders günstiges Pruning möglich würde — dann wäre dieses Kriterium aber wahrscheinlich zu schwach, um nützlich zu sein. Zum Schluss sei noch bemerkt: Alle Merkmalsuchverfahren, die auf der Idee der Greedy-vorwärts-Suche basieren, sind im Wesentlichen Algorithmen für Entscheidungsbaumkonstruktion ohne Pruning. Eine der Fragen, die sich dabei stellen, ist es, ob man — im Falle einer Klassifizierung durch Entscheidungsbäume — dadurch etwas gewinnt, dass für Merkmalsuche andere Kriterien angewandt werden als für die Klassifizierung. Nach dem Wissen des Autors gibt es dazu in der Literatur keine Überlegungen.

1.1.3 Skalarprodukt-Score

Man kann versuchen, die zwei zentralen Desiderate der effizienten Merkmalsuche, das heißt große Korrelation mit der Zielvariable und kleine Korrelation untereinander direkt angehen und eine Score-Funktion konstruieren, die die Tauglichkeit einer gegebenen Merkmalmenge mithilfe eines Skalarproduktes evaluiert. Die Score-Funktion berechnet die Tauglichkeit neuer Merkmale bezüglich der schon in früheren Schritten des Algorithmus ausgewählten. Sie kann wie folgt definiert werden [68].

$$\text{score}(\mathbf{s}_{\text{new}}) = - \sum_{\mathbf{s} \in \mathcal{S}'} GV(\mathbf{s}_{\text{new}}) \cdot GV(\mathbf{s}) + (|\mathbf{s}_{\text{new}}| + \beta GV(\mathbf{s}_{\text{new}}))|_{\mathcal{L}} \cdot TV$$

\mathcal{S}' ist hier die Menge der Graphen, die schon in früheren Schritten ausgewählt wurden, \mathcal{L} ist die Menge der etikettierten Graphen und das Skalarprodukt \cdot entspricht dem üblichen Skalarprodukt auf reellen Zahlen, der auf Vektoren mit 1 und -1 ausgeführt wird. Die Normierungskonstante β , kann entweder als Parameter betrachtet und beliebig gesetzt werden; man kann auch $\beta = (1 + |UL|/|\mathcal{L}|)$ setzen, so dass die zwei Arten von Korrelation gleichermaßen zur Geltung kommen. Das Ziel des Verfahrens ist es, solche Merkmale zu finden, die möglichst große Werte der Score-Funktion erzeugen.

Die hier vorgestellte Variante eignet sich für binäre Klassifizierung, man könnte sie aber durch Modifizierung oder Ersetzen des letzten Summands der Score-Funktion relativ leicht auf Kategorisierung oder Regression verallgemeinern (wobei bei Regression wahrscheinlich eine andere Definition der Skalarproduktes nötig wäre).

1.1.4 Submodulare Inkonsistenzenfunktion

Ein weiterer Ansatz basiert auf dem Begriff der submodularen Funktion [84].

Definition 8. Eine Funktion $u : 2^{\mathcal{A}} \rightarrow \mathbb{R}$ ist dann SUBMODULAR auf \mathcal{A} , wenn für alle $\mathcal{S}' \subset \mathcal{S} \subseteq \mathcal{A}$ und $\mathbf{x} \in \mathcal{A}$ gilt, dass $u(\mathcal{S}' \cup \{\mathbf{x}\}) - u(\mathcal{S}') \geq u(\mathcal{S} \cup \{\mathbf{x}\}) - u(\mathcal{S})$.

In anderen Worten bedeutet dies, dass das Hinzufügen eines Merkmals in einem früheren Schritt des Greedy-Algorithmus zumindest so viel bringt wie in einem späteren. Diese Klasse von Funktionen ist wegen des nachfolgenden Theorems interessant.

Theorem. Wenn u submodular und nichtabsteigend ist und $u(\emptyset) = 0$, so gilt Folgendes für die Ausgabe \mathcal{S}' der Greedy-Vorwärts-Suche: $u(\mathcal{S}') \geq (1 - 1/e) \max_{\mathcal{U} \subseteq \mathcal{A}, |\mathcal{U}|=|\mathcal{S}'|} u(\mathcal{U})$.

Der Beweis befindet sich in [61]. Das Theorem bedeutet, dass der Greedy-Ansatz, der von der Komplexität her nur linear ist in der Anzahl der auszuwählenden Merkmale im pessimistischen Fall nur $1/e$, also ungefähr 37% schlechter ist, als ein Algorithmus, der alle Untermengen von \mathcal{A} betrachten würde und somit exponentielle Komplexität hätte.

Nun gilt es also, eine geeignete Funktion u zu definieren. Eine Möglichkeit ist es, den Wert von $u(\mathcal{A})$ von der Anzahl der Inkonsistenzen in der Menge \mathcal{A} abhängig zu machen. Eine Inkonsistenz ist eine Situation, wo die gleiche Konfiguration von Merkmalbelegungen unterschiedliche Ausgaben erzeugt. Für Klassifizierungsprobleme können wir sie so definieren: Ein Paar der Moleküle $\mathbf{g}_i, \mathbf{g}_j$ bildet bezüglich der Merkmalmenge \mathcal{M} dann eine Inkonsistenz, wenn $l(\mathbf{g}_i) \neq l(\mathbf{g}_j)$ und $MV(\mathbf{g}_i) = MV(\mathbf{g}_j)$. Die Score-Funktion kann nun für binäre Klassifizierungsprobleme definiert werden.

$$\text{score}(\mathbf{s}_{\text{new}}) = -(\text{Anzahl der Inkonsistenzen bezüglich } \mathcal{S}' \cup \{\mathbf{s}_{\text{new}}\}) + |\mathcal{G}_0| |\mathcal{G}_1|$$

Die Symbole $|\mathcal{G}_0|$ und $|\mathcal{G}_1|$ stehen hier für die Anzahl der Graphen mit jeweils positiver oder negativer Klassifizierung. Es ist einfach festzustellen, dass diese Score-Funktion tatsächlich eine submodular ist. Ein Vorteil dieser Definition ist es, dass die Greedy-Suche in die Graphenextraktion eingebunden werden kann und dass es Pruning-Kriterien gibt, die die Extraktion beschleunigen. Sie werden hier nicht vorgestellt, Details dazu sind bei Borgwardt et. al. [13] sowie Thoma et. al. [84] zu finden.

Die implementationstechnische Frage, die man sich jetzt stellt, ist es, wie die Anzahl der Inkonsistenzen effizient berechnet werden kann. Ein möglicher Ansatz ist es, die Graphen als Bitvektoren mit den Merkmalen aus der gerade betrachteten Menge der Merkmale \mathcal{S}' darzustellen. Dann geht man die sortierte Liste der so kodierten Graphen der Reihe nach entlang und zählt die Inkonsistenzen, indem man nach der Konfiguration sucht, wo mehrere identische Vektoren nacheinander folgen. Unter den Graphen, die diese Vektoren repräsentieren ermittelt man dann die Zahl deren mit positiver ($|\mathcal{G}_1^v|$) und negativer ($|\mathcal{G}_0^v|$) Klassifizierung und fügt $|\mathcal{G}_1^v| |\mathcal{G}_0^v|$ zu dem Inkonsistenzanzähler hinzu. Wo zwei nachfolgende Vektoren unterschiedlich sind, können die entsprechenden Graphen aus der Liste entfernt werden. Alternativ dazu kann man auch eine TRIE-Baumstruktur konstruieren, in der die Kanten den Merkmalen entsprechen, wobei die Kanten, die nah an den Wurzeln sind, für die zuerst hinzugefügten Merkmale stehen, und wo die Blätter für Graphenmengen stehen. Die Gesamtanzahl der Inkonsistenzen ergibt sich dann aus der Summe der Inkonsistenzen für einzelne Blätter, die ähnlich wie in der Listenmethode ermittelt wird. Der Algorithmus wird hier nicht in Pseudocode vorgestellt. Als Stoppbedingung betrachtet man die Situation, wo der nachfolgende Schritt die Anzahl der Inkonsistenzen nicht weiter verringert. Alternativ kann man auch früher, nach einer festgelegten Anzahl der Iterationen, terminieren.

Dieser Ansatz ist mit Boosting (wo man versucht, dass nachfolgende Klassifizierer bestehende Konflikte auflösen) sowie mit der Konstruktion eines Entscheidungsbaumes nach

score verwandt. Der Hauptnachteil ist es, dass die Definition von *score* sehr einfältig ist, was zur Folge hat, dass die unetikettierten Graphen nicht benutzt werden können. Eine ausführliche Beschreibung der Ansätze, die auf (In-)Konsistenzen basieren, geben Dash et. al. [28]. Konzeptuell gesehen können sie auch als eine Variante der früheren Trenne-und-herrsche-Ansätze im Regellernen angesehen werden (wie sie in [35] beschrieben werden), wo man iterativ neue Regeln hinzufügt, so dass die Klassen der Zielvariablen voneinander separiert werden. Es bleibt zu erforschen, ob es submodulare Funktionen gibt, die ebenfalls die nicht etikettierten Graphen als Informationsquelle nutzen können. Ein weiteres Problem ist es, dass man sie Untergraphenstruktur lediglich zur Optimierung (Pruning) benutzen kann, aber nicht zur Verbesserung der Filterungsqualität, Es ist auch nicht sofort ersichtlich, wie submodulare Funktionen konstruiert werden können, die sich für Regressionsprobleme eignen. Ein gutes Merkmal des Algorithmus dagegen ist es, dass das Score-Kriterium auf eine sehr kompakte Art und Weise sowohl die Korrelation der Merkmale mit der Zielvariable als auch untereinander misst.

1.1.5 Wrappers

Eine noch andere Möglichkeit, Merkmalsuche durchzuführen, ist es, die Relevanz verschiedener Attribute nicht mithilfe eines eigens konstruierten Maßes zu evaluieren, sondern direkt auf einen Klassifizierer zuzugreifen. Dieser Klassifizierer kann nun entweder der gleiche sein, der später benutzt wird, um Prognosen durchzuführen, oder ein anderer. Ein häufiger Ansatz ist es, einen einfacheren Klassifizierer für die Merkmalsuche als für den späteren Genrealisierungsschritt anzuwenden, da dies einerseits die Merkmalsuche beschleunigt und andererseits hilft, der Überanpassung vorzubeugen — zum Beispiel nimmt man jeweils eine SVM mit linearem Kernel und eine mit einem komplizierteren. Nun können wir also den eigentlichen Ansatz definieren. Es bleibt also nun, die neue Score-Funktion zu definieren.

$$score(\mathbf{s}_{new}) = Acc^C(\{\mathbf{s}_{new}\} \cup \mathcal{S}, T) - \beta Acc^{C_{bin}}(\mathcal{S}, \mathbf{s}_{new})$$

Dabei bedeutet der Ausdruck $Acc^C(\mathcal{M}, Z)$ die Trefferquote des Klassifizierers C , wobei als Eingabe die Merkmale aus \mathcal{M} dienen und Z die vorauszusagende Variable ist, T bedeutet die globale Zielvariable (das Etikett des jeweiligen Graphen), β ist ein Balanzierkoeffizient. Dabei muss die Menge der Beispiele jeweils in eine Test- und Trainingsmenge unterteilt sein. Diese Aufteilung erfolgt in einer inneren Schleife des Merkmalsuchalgorithmus und ist nicht die gleiche Aufteilung, die zur Evaluierung der globalen Kette der Merkmalsuche und des Lernens benutzt wird. Es stehen dem Merkmalsuchverfahren ja nur diese Objekte zur Verfügung, die sich in der Trainingsmenge der äußeren Schleife befinden. Wenn die Etiketten Boolesche Werte sind, so kann man $C_{bin} = C$ setzen, ansonsten muss man grundsätzlich zwei Klassifizierer definieren: einen, der mit den Etiketten arbeitet, und benutzt wird, um die Relevanz eines gewissen Merkmals für die Zielvariable zu messen und einen Anderen, der Boolesche Werte ausgibt und die Korrelationen der Merkmale untereinander misst. Es ist dabei wichtig zu erkennen, dass der zweite keine Etiketten braucht und deswegen zum Trainieren auch unmarkierte Graphen benutzt werden können.

Eine wichtige Frage, die man sich bei allen Wrapper-Ansätzen stellen soll, ist die Art und Weise, wie die Trefferquote ermittelt wird. Man hat dabei meistens die folgende Wahl: Entweder lässt man den Klassifizierer nur einmal laufen und nimmt die Zahl oder man benutzt statistische Techniken um bei mehreren Durchläufen, von denen man annimmt, dass sie unabhängig sind, das Intervall, wo sich die tatsächliche Präzision befindet, bezüglich eines gewissen Vertrauensbereiches zu ermitteln, wobei die Durchläufe so lange durchgeführt werden, bis das Intervall klein genug ist. Eine heuristische Variante davon ist es, eine Schranke für die Standardabweichung zu setzen (was aber nicht theoretisch zu begründen ist). Man

muss dabei natürlich auch den Fall beachten, dass ein Klassifizierer unter Umständen Trefferquoten aufweisen kann, die breit zerstreut sind und keinen erkennbaren Grenzwert erkennen lassen — dies zeugt zumeist von einem schlechten Entwurf des Klassifizierers. So oder so stellt sich hier ein Dilemma, ob man Rechnerzeit eher dafür aufwenden soll, ein genaues Maß der jetzigen Merkmalmenge zu berechnen (das entspricht der Situation, wo man ein schmales Intervall verlangt) oder mehr Mengen zu untersuchen. In der Literatur wird dies das Exploitation-vs-Exploration-Problem genannt [47]. Natürlich gibt es hier keine universell gute Lösung, weil die Antwort von der Natur der Suchheuristik und auch der betrachteten Datenmenge abhängig ist. Wrapper-Ansätze lassen sich natürlich auch für Regression definieren.

Es sei hier bemerkt, dass die Unterscheidung, ob ein Ansatz ein Wrapper-Ansatz ist oder nicht grundsätzlich Konventionssache ist. So kann man sich im Prinzip auch z.B. für klassische Filteransätze wie χ^2 die Definition eines künstlichen χ^2 -Klassifizierers vorstellen, so dass der Filteransatz als Wrapper-Ansatz angesehen werden kann. Das Entscheidende dabei ist es, welche Betrachtungsweise in einem bestimmten Fall die natürliche ist — wenn der Klassifizierer eher kompliziert ist, ist das eher der Wrapper-Ansatz.

Eine andere in der Literatur anzutreffende Variante von Wrapper-Methoden ist die eingebettete Merkmalsuche — ein Wrapper-Ansatz wird dann als eingebettet beschrieben, wenn man den eingebauten Klassifizierer nicht mehr als eine Black-Box betrachtet, wo man nur die Trefferquote berücksichtigt, sondern auch versucht, seine innere Struktur bei der Suche zu berücksichtigen, z.B. Gradientenabstieg bezüglich der Fehlerfunktion des Klassifizierers macht. Diese Art von Methoden wurde hier nicht untersucht.

1.1.6 Markovsche Decke

Ein anderer Ansatz der Merkmalsuche basiert auf dem Begriff der Markovschen Decke [49].

Definition 9. Die MARKOVSCHE DECKE für ein Merkmal \mathbf{f}_i ist eine Menge $\mathcal{M}_i \subseteq \mathcal{S}$, so dass $I(\{\mathbf{f}_i\}, T \cup \mathcal{S} \setminus \{\mathbf{f}_i\} \setminus \mathcal{M}_i | \mathcal{M}_i)$, wobei I für bedingte Unabhängigkeit steht.

Dieser Begriff ist deshalb für die Merkmalsuche nützlich, weil die Markovschen Decken dank des nachfolgenden Theorems mit Greedy-Rückwärtssuche kompatibel sind, das heißt, die Greedy-Suche ist optimal, wenn wir Merkmale eliminieren aber die entsprechenden Decken in der Merkmalmenge belassen.

Bevor wir das Theorem beweisen, ist es nötig, drei Eigenschaften von bedingter Unabhängigkeit einzuführen, die für den Beweis nötig sind.

$$\begin{array}{ll}
 I(\mathcal{A}, \mathcal{B} \cup \mathcal{C} | \mathcal{D}) \Rightarrow I(\mathcal{A}, \mathcal{B} | \mathcal{D}) & \text{Dekomposition} \\
 I(\mathcal{A}, \mathcal{B} \cup \mathcal{C} | \mathcal{D}) \Rightarrow I(\mathcal{A}, \mathcal{B} | \mathcal{D} \cup \mathcal{C}) & \text{Schwache Vereinigung} \\
 I(\mathcal{A}, \mathcal{B} | \mathcal{C} \cup \mathcal{D}) \wedge I(\mathcal{A}, \mathcal{D} | \mathcal{C}) \Rightarrow I(\mathcal{A}, \mathcal{D} \cup \mathcal{B} | \mathcal{C}) & \text{Kontraktion}
 \end{array}$$

Diese Eigenschaften können durch die Anwendung der Definition von bedingter Unabhängigkeit bewiesen werden, jetzt wird darauf verzichtet.

Theorem. Wenn $\mathbf{f}_j \in \mathcal{G}$ ein Merkmal ist, das wir gerade eliminieren und $\mathbf{f}_i \notin \mathcal{G}$ ein Merkmal ist, das in einem früheren Schritt aufgrund der Decke \mathcal{M}_i eliminiert worden ist, so hat das Merkmal \mathbf{f}_i eine Decke in $\mathcal{G} \setminus \{\mathbf{f}_j\}$.

Beweis. Wenn $\mathbf{f}_j \notin \mathcal{M}_i$, so ist das Theorem trivial erfüllt, da wir einfach die alte Decke übernehmen können. Im Falle, dass $\mathbf{f}_j \in \mathcal{M}_i$ müssen wir eine neue Decke konstruieren. Wir werden jetzt beweisen, dass $\mathcal{M}_i \setminus \{\mathbf{f}_j\} \cup \mathcal{M}_j$ eine Decke für \mathbf{f}_i ist. Wir führen jetzt zur

Vereinfachung der Schreibweise die Mengen $\mathcal{M}'_i = \mathcal{M}_i \setminus \{\mathbf{f}_j\}$ und $\mathcal{X} = T \cup (\mathcal{G} \setminus \{\mathbf{f}_j\}) \setminus \mathcal{M}'_i \setminus \mathcal{M}_j$ ein. Das Ziel ist es nun folgendes zu Beweisen.

$$I(\{\mathbf{f}_i\}, \mathcal{X} | \mathcal{M}'_i \cup \mathcal{M}_j)$$

Es sei hier bemerkt, dass wir aus dem Grunde nicht $I(\{\mathbf{f}_i\}, \mathcal{X} \setminus \{\mathbf{f}_i\} | \mathcal{M}'_i \cup \mathcal{M}_j)$ bewiesen müssen, weil $\mathbf{f}_i \notin \mathcal{G}$. Wichtig ist auch zu beachten, dass in \mathcal{X} der Ausdruck $(\mathcal{G} \setminus \{\mathbf{f}_j\})$ statt \mathcal{G} vorkommt, dies entspricht der Tatsache, dass sich die Merkmalmenge nach dem Ausführen des aktuellen Schrittes um \mathbf{f}_j verkleinert.

Da \mathcal{M}_j eine Markovsche Decke für das Merkmal \mathbf{f}_j ist, wissen wir, dass $I(\{\mathbf{f}_j\}, \{T\} \cup \mathcal{G} \setminus \mathbf{f}_j \setminus \mathcal{M}_j | \mathcal{M}_j)$ gilt. Dies lässt sich als $I(\{\mathbf{f}_j\}, \mathcal{X} \cup \mathcal{M}'_i | \mathcal{M}_j)$ umschreiben, woraus wegen der Eigenschaft der schwachen Vereinigung (1) $I(\{\mathbf{f}_j\}, \mathcal{X} | \mathcal{M}_j \cup \mathcal{M}'_i)$ folgt. Ähnlich kann man bei dem anderen Merkmal argumentieren: Da \mathcal{M}_i eine Markovsche Decke für das Merkmal \mathbf{f}_i ist und unter Benutzung der Eigenschaft der schwachen Vereinigung (da seitdem das Merkmal \mathbf{f}_i eliminiert wurde die Menge der Merkmale auf \mathcal{G} reduziert wurde) können wir sagen, dass $I(\{\mathbf{f}_i\}, \{T\} \cup \mathcal{G} \setminus \mathcal{M}_i | \mathcal{M}_i)$ gilt. Das lässt sich als $I(\{\mathbf{f}_i\}, \mathcal{X} \cup \mathcal{M}_j | \mathcal{M}_i)$ umschreiben, was wegen der Eigenschaft der schwachen Vereinigung $I(\mathbf{f}_i, \mathcal{X} | \mathcal{M}_i \cup \mathcal{M}_j)$, also (2) $I(\{\mathbf{f}_i\}, \mathcal{X} | \mathcal{M}'_i \cup \mathcal{M}_j \cup \{\mathbf{f}_j\})$ nach sich zieht.

Aus (1) und (2) folgt nun aufgrund der Eigenschaft der Kontraktion ($\mathcal{A} = \mathcal{X}$, $\mathcal{B} = \{\mathbf{f}_i\}$, $\mathcal{C} = \mathcal{M}'_i \cup \mathcal{M}_j$, $\mathcal{D} = \{\mathbf{f}_j\}$), dass $I(\{\mathbf{f}_i\} \cup \{\mathbf{f}_j\}, \mathcal{X} | \mathcal{M}_j \cup \mathcal{M}'_i)$ gilt. Aufgrund der Eigenschaft der Dekomposition impliziert dies die Zielformel. \square

Praktisch ist es nötig, die Decken approximiert zu berechnen, da eine genaue Berechnung der bedingten Wahrscheinlichkeiten viel zu zeitaufwändig wäre. Ein möglicher Ansatz dafür sieht wie folgt aus: Man geht von einer konstanten Größe der Decke aus (meistens 1 oder 2) und wählt für die Decke die Merkmale aus, die mit dem betrachteten Merkmal im Hinblick auf die Zielvariable am wenigsten korreliert sind (die Korrelationen werden für jedes Paar im Voraus berechnet). Dies ist nur unter der vereinfachenden Annahme sinnvoll, dass sich die Korrelationen zwischen den Merkmalen dann am stärksten ausprägen, wenn sie direkt sind und keine anderen Merkmale dazwischen sind. Auf diese Weise erzeugt man für jedes noch erhaltene Merkmal eine Kandidatendecke. Diese wird dann evaluiert und es wird das Merkmal entfernt, das mit seiner Decke am wenigsten Korreliert ist. Bevor wird diesen Vorgang formell niederschreiben können, werden wir die Korrelationsmaße definieren.

Zuerst werden die paarweisen Korrelationen nach der folgenden Formel berechnet.

$$D_{i,j}^{b_1,b_2} = \sum_{c \in \{0,1\}} P(T = c | \mathbf{s}_i = b_1, \mathbf{s}_j = b_2) \log \frac{P(T = c | \mathbf{s}_i = b_1, \mathbf{s}_j = b_2)}{P(T = c | \mathbf{s}_j = b_2)}$$

$$\gamma_{i,j} = \sum_{b_1, b_2 \in \{00,01,10,11\}} P_{UL}(\mathbf{s}_i = b_1, \mathbf{s}_j = b_2) D_{i,j}^{b_1,b_2}$$

Die Wahrscheinlichkeiten werden mit Frequenzen approximiert. Die Notation $\mathbf{s}_j = 1$ bedeutet, dass das i -te Merkmal in dem betrachteten Graphen vorhanden ist. Die unetikettierten Graphen können in die Berechnung von P_{UL} direkt miteinbezogen werden, die Berechnung von P kann nur unter Ausnutzung der etikettierten Graphen erfolgen (eventuell mit Smoothing). Es ist wichtig, zu bemerken, dass die Korrelationen nicht symmetrisch sind, intuitiv kann das damit begründet werden, dass $\gamma_{i,j}$ den Wissensgewinn repräsentiert, den \mathbf{s}_j im Hinblick auf das jeweilige Paar bringt, was eine asymmetrische Eigenschaft ist.

Die errechneten Decken werden mithilfe einer ähnlichen Formel evaluiert.

$$D_{M_i,j}^{b_1,b_2,\dots,b_k,b_{\mathbf{f}_j}} = \sum_{c \in \{0,1\}} P(T = c | \mathcal{M}_i = (b_1, \dots, b_k), \mathbf{f}_j = b_{\mathbf{f}_j})$$

$$\log \frac{P(T = c | \mathcal{M}_i = (b_1, \dots, b_k), \mathbf{s}_j = b_{\mathbf{s}_j})}{P(T = c | \mathcal{M}_i = (b_1, \dots, b_k))}$$

$$\delta_i = \sum_{b_1, b_2, \dots, b_k, b_{s_j} \in \text{Bin}_{k+1}} P_{UL}(\mathcal{M}_i = (b_1, \dots, b_k), \mathbf{s}_j = b_{s_j}) D_{\mathcal{M}_i, j}^{b_1, b_2, \dots, b_k, b_{s_j}}$$

Die Größe der Decke ist hier als k markiert. Bin_{k+1} steht für die Menge aller binären Wörter der Länge $k + 1$. Der Ausdruck $\mathcal{M}_i = (b_1, \dots, b_k)$ bedeutet, dass alle Merkmale, die in der Decke \mathcal{M}_i sind, jeweils den Wert $b_1, \dots, b_k \in \{0, 1\}$ haben (das heißt die Untergraphen sind entweder da oder nicht). Auch hier können in die Berechnung von P_{UL} die unetikettierten Graphen miteinbezogen werden. Der gesamte Algorithmus sieht nun so aus (Algorithmus 2). Alternativ kann man als Stoppbedingung den kleinsten δ -Wert nehmen (stoppen, wenn er zu groß wird).

1.1.7 Vergleich der Ansätze

In diesem Abschnitt werden die oben beschriebenen Ansätze zur Merkmalsuche verglichen. Es handelt sich hier lediglich um einen theoretischen Vergleich, praktische Ergebnisse werden im Kapitel 3 erläutert. Einen Überblick bietet die Tabelle 1.1. Man erkennt sofort, dass der Wrapper-Ansatz am universellsten erscheint, was insofern wenig verwunderlich ist, als man dank der Benutzung der eingebauten Klassifizierer den schwierigsten Teil der Definition eines Merkmalsuchverfahrens — die Definition der Korrelationen zwischen den Merkmalen — vermeiden konnte. Der entscheidende Nachteil ist die lange Ausführungszeit — dies ist nicht zu vermeiden, da man beim Hinzufügen jedes Merkmals in die Menge \mathcal{S}' das Trainingsverfahren für den Klassifizierer $O(|\mathcal{S}'|)$ -mal Aufrufen muss. Es kann zwar sein, dass dies für einige Klassifizierer gemindert werden kann, indem man iterative Trainingsverfahren anwendet, die die Operation des Hinzufügens und des Entfernens eines Trainingspunktes unterstützen. Da dies aber a priori die Auswahl eines bestimmten Klassifizierers voraussetzt, wurde in dieser Arbeit darauf verzichtet.

Die zwei Verfahren, die für die Bewertung der Zweckmäßigkeit von teilüberwachter Merkmalsuche am ehesten geeignet sind, sind die Markovsche Decke und das Skalarprodukt: Sie haben akzeptable Ausführungszeiten und die unetikettierten Daten werden auf eine nicht triviale Art und Weise genutzt, was einen Zuwachs der Klassifizierungsperformanz zumindest erhoffen lässt. Der Inkonsistenzen-Score ist in der vorgestellten Version durch die Score-Funktion begrenzt. Das Verfahren wird erst interessanter, wenn man eine neue submodulare Funktion, die auch die unetikettierten Daten miteinbeziehen würde. Es sind dem Autor keine

Algorithmus 2 Merkmalsuche mit Markovschen Decken.

Markovsche-Decke(k)

begin

$\mathcal{S}' = \mathcal{S}$

 while $|\mathcal{S}'| > \text{anzahlMerkmale}$ do

schlechtestesMerkmal = ein beliebiges Merkmal aus \mathcal{S}'

 foreach $\mathbf{f}_i \in \mathcal{S}'$ do

$M_i \leftarrow k$ Merkmale aus \mathcal{S}' mit dem kleinsten $\gamma_{i,j}$

 if $\delta_i < \delta_{\text{schlechtestesMerkmal}}$

schlechtestesMerkmal = \mathbf{f}_i

 fi

 od

 od

$\mathcal{S}' = \mathcal{S}' \setminus \text{schlechtestesMerkmal}$

end

Tabelle 1.1 Vergleich verschiedener Merkmalsuchverfahren.

	χ^2	χ^2 m. Support	Markov	Wrapper	Skalarprodukt	Inkonsistenzen
Teilüberwachtes Lernen	✗	✓	✓	✓	✓	✗
Korr. untereinander	✗	✗	✓	✓	✓	✓
Korr. mit Target	✓	✓	✓	✓	✓	✓
Gitterstruktur nötig	✗ ^a	✗ ^a	✗ ^a	✓	✓	✗
Einbindbar in Extraktion ^b	✓	✓	✗	✓ ^c	✓ ^c	✓ ^c
Zeitbedarf (theoretisch $O(\cdot)$) ^d	kn	kn	k^2n	k^2g^e	k^3n^f	k^2n
Zeitbedarf (Sek., gemessen ^g)	< 1	< 1	15	67	< 1	< 1

^a Die Gitterstruktur kann zur Optimierung der Extraktion benutzt werden, hat aber keinen Einfluss auf das Ergebnis der Merkmalsuche.

^b Die negativen Angaben in dieser Zeile bedeuten, dass es nicht trivial ist, und dass dem Autor keine Ansätze dazu bekannt sind; es wird nicht unterstellt, dass es unmöglich ist.

^c Nur bei Suchheuristik, die der Nachbarsuche im Gitter \subseteq entspricht.

^d Symbole: n -Anzahl der Graphen, k -Anzahl der Attribute.

^e Mit $g = (k, n)$ wurde die Komplexität des benutzten Klassifizierers bezeichnet.

^f Bei Greedy-Suche.

^g Die Messung wurde mit einer zufälligen, aber für alle Ansätze gleichen Untermenge (20%) des FONTAINE-Datensatzes durchgeführt. Es wurde 10-fache 10-Fold Kreuzvalidierung gemacht, die Zeitwerte wurden gemittelt, die Ergebnisse ignoriert..

solche Funktionen bekannt. Die anderen Ansätze (insbesondere HKA) dienen dem Vergleich mit lang etablierten, standardmäßigen Vorgehensweisen.

1.2 Verfahren für Regression

Die vorher beschriebenen Ansätze sind, außer Wrapper, nicht direkt für Regression geeignet. In diesem Abschnitt werden zwei Verfahren vorgestellt, die von vornherein auch für kontinuierliche Variablen gedacht sind.

1.2.1 Pearson-Korrelation

Zuerst stellen wir ein Baseline-Verfahren vor, das ungefähr dem χ -Quadrat-Ansatz für Klassifizierung entspricht: Wir berechnen für jedes Merkmal eine Score-Funktion, die nur die Korrelation mit der Zielvariable beschreibt, aber nicht mit anderen Merkmalen. Eine einfache Möglichkeit, dies zu tun, bietet der Pearson-Koeffizient. Er hat den Nachteil, dass er lediglich die lineare Korrelation zwischen zwei Variablen beliebiger Verteilung misst (für normalverteilte Variablen sagt er alles über die Abhängigkeitsrelation aus, bei unserer Aufgabe sind aber die Variablen nicht unbedingt normalverteilt). Wir haben aber bereits bei der Beschreibung der Merkmalsuche festgestellt, dass es unmöglich ist, völlig allgemein Korrelation zu identifizieren, ähnlich wie es unmöglich ist, einen Klassifizierer ohne Bias zu bauen. Die Pearson-Korrelation ist wie folgt definiert. Als Score-Funktion wird das Quadrat dieser Größe verwendet.

$$Cor(\mathbf{x}, T) = \frac{\text{Cov}(\mathbf{x}, T)}{\sigma(\mathbf{x})\sigma(T)}$$

1.2.2 Mutual Information

Eine andere mögliche Score-Funktion ist der Mutual-Information-Wert, der in der Literatur zur Merkmalsuche breite Verwendung findet (siehe z.B. [87], [11], [3]). Im Falle, wo die Attribute binär sind und die Zielvariable kontinuierlich ist, wird er folgendermaßen definiert¹.

$$I(T, \mathbf{s}) = \sum_{c \in \{0,1\}} \int_{\mathbb{R}} P(\mathbf{s} = c, T = t) \log \frac{P(\mathbf{s} = c, T = t)}{P(T = t)P(\mathbf{s} = c)} dt = \\ \sum_{c \in \{0,1\}} \int_{\mathbb{R}} P(T = t | \mathbf{s} = c) P(\mathbf{s} = c) \log \frac{P(T = t | \mathbf{s} = c)}{P(T = t)} dt$$

Die Wert $P(\mathbf{s} = c)$ ist dabei unabhängig von t und kann gut durch die relative Häufigkeit des betrachteten Untergraphen approximiert werden. Die Dichten $P(T = t | \mathbf{s} = 0)$, $P(T = t | \mathbf{s} = 1)$ und $P(T = t)$ müssen dagegen durch kompliziertere Methoden approximiert werden, da sie von einer kontinuierlichen Variable abhängig sind. Da man im Allgemeinen über die Natur dieser Verteilungen nichts annehmen kann, ist es zweckmäßig, nichtparametrische Methoden anzuwenden [11]. Es sei hier bemerkt, dass die drei dichten nur eindimensional sind, da wir keine Korrelationen zwischen den Merkmalen berücksichtigen. Die gängige Methode für die nichtparametrische Dichtenapproximation sind Parzen-Fenster [30]. Gegeben sei eine Menge von Objekten, der zu modellierenden Verteilung entsprechen – also z.B. für $P(T = t | \mathbf{s} = 0)$ die Etiketten all den Graphen, die den betrachteten Untergraphen nicht haben. Nehmen wir an, dass diese Etiketten y_1, \dots, y_k sind. Dann ist die Dichte durch die folgende Formel gegeben.

$$P(T = t | \mathbf{s} = 0) = \frac{1}{kh} \sum_{i=1}^{i=k} K \left(\frac{t - y_i}{h} \right)$$

Die (vom Benutzer gesetzte) konstante h (die Breite des Fensters) entspricht hier der Glätte der Approximation. Die Kernel-Funktion K wurde in dieser Arbeit auf die Normalverteilung gesetzt: $K(x) = (\sqrt{2\pi})^{-1} e^{-\frac{1}{2}x^2}$. Der einzige verbleibende Schritt ist noch die Berechnung der Integrale. Hier wurde eine einfache Trapezoid-Quadratur verwendet, die den Argumentenraum solange in zwei Teile spaltet, bis der Wert der Integrale stabil wird. Natürlich wäre es möglich, ein komplizierteres Verfahren anzuwenden oder statt einem allgemeinen Verfahren eins zu konstruieren, das gezielt auf die Struktur der Parzen-Dichtenapproximation abgestimmt ist. Da die Berechnung der Integrale hier aber weder konzeptuell noch von der Komplexität her der wichtigste Schritt ist, wurde darauf verzichtet. Es wäre auch möglich, den Parameter h nicht konstant zu setzen, sondern in verschiedenen Regionen der Dichte andere Fensterbreiten zu benutzen (siehe [83]). Auch das wurde hier nicht verfolgt.

1.2.3 Rayleigh-Koeffizient

Ein anderer Ansatz basiert auf dem Rayleigh-Koeffizienten. Man definiert zwei symmetrische Matrizen N und C gleicher Größe, die jeweils für die gewünschte und unerwünschte Eigenschaften der Eingabe stehen (Information vs. Rauschen). Wir suchen nun die ersten k Vektoren w_1, \dots, w_k , so dass sie C -orthogonal sind (d.h. $w_i^\top C w_j = 0$ für $i \neq j$) und der folgende Ausdruck möglichst groß ist [58].

$$J(w) = \frac{w^\top N w}{w^\top C w}$$

¹In den Arbeiten [11], [3] wird, eine kompliziertere Variante verwendet, wo man auch Korrelationen zwischen den Merkmalen misst: paarweise in [3] und für die ganze gewählte Menge in [11].

Eine allgemeine Methode, dieses Problem zu lösen, bietet die Umwandlung in das generalisierte Eigenwert-Problem²: Man finde die Paare w und λ , so dass $Nw = \lambda Cw$. Die Details dieser Umwandlung befinden sich in [22] (siehe Seite 131). Wenn wir zusätzlich annehmen, dass die Matrizen C, N symmetrisch positiv semidefinit sind, kann nachgewiesen werden, dass alle Eigenwerte reell sind (siehe Fix et. al.[33])³. Davon wählen wir die k mit den größten λ -Werten aus. Eine genauere Besprechung des allgemeinen Falls des generalisierten Eigenwert-Problems befindet sich bei Moler et. al [59]. In dieser Arbeit wurde zusätzlich angenommen, dass der Nullraum von N den Nullraum von C beinhaltet. Das ermöglicht die Reduktion des Problems auf die Berechnung von einfachen Eigenvektoren für eine Matrix. In der Tat, betrachten wir die Diagonalisierung $C = Q\Lambda Q^\top$ (wir nehmen an, dass die Eigenwerte absteigend sortiert sind und dass es p Eigenwerte gibt, die nicht nah an null sind). Dann entspricht die Minimierung von $w^\top Nw/w^\top Cw$ der Minimierung von $(Qc)^\top N(Qc)/(Qc)^\top C(Qc) = c^\top Q^\top NQw/c^\top \Lambda c$, wobei c ein Vektor in der Basis Q ist. Jetzt können wir die Elemente von c , die den Null-Eigenwerten von C entsprechen, auf null fixieren, da sie wegen der Nullraum-Annahme den Wert des Koeffizienten nicht beeinflussen. Jetzt können wir statt der Matrizen $Q^\top NQ$ und Λ die entsprechenden Minoren-Matrizen nehmen, die die Größe $p \times p$ haben und die in den oberen linken Ecken von $Q^\top NQ$ und Λ anfangen. Bezeichnen wir diese als N' und Λ' . Nun lösen wir $N'c = \lambda \Lambda'c$. Λ' ist aber nicht-singulär. Deswegen reicht es, die Eigenvektoren c_i der (nicht unbedingt symmetrischen) Matrix $\Lambda'^{-1}N'$ zu betrachten. Diese können dann mit $w_i = Qc'_i$ in die kanonische Basis transformiert werden, wobei c' der Vektor c ist, der um $k - p$ Nullen unten erweitert wurde. Zur Berechnung der Eigenvektoren wurde in dieser Arbeit die Lapack-Bibliothek verwendet.

Die entscheidende Frage ist natürlich, wie wir die Matrizen N, C definieren. Pan et. al. [65] schlagen vor, die bereits beschriebene Pearson-Korrelation zu nehmen. Machen wir jetzt auch einige Normierungsannahmen: Bezeichnen wir die normierte Version des Graphenvektors $GV(\mathbf{g}_i)$ als x_i (x_1 bis x_l sind etikettiert, x_{l+1} bis x_n haben keine Etiketten). Die Normierung bedeutet, dass $\sum_i^n x_i = 0$. Nehmen wir auch an, dass $\sum_i y_i = 0$ und $\sum_i y_i^2 = 1$ (wir können das immer durch eine lineare Transformation der Etiketten erreichen). Betrachten wir nun zwei Zufallsvariablen: das neu generierte Merkmal $w^\top[x_1|\dots|x_n]$ und die Zielvariable t . Es kann einfach überprüft werden, das folgendes gilt.

$$\begin{aligned} \text{Cor}^2([x_1|\dots|x_n]^\top w, [y_1, \dots, y_l, y_{l+1}^*, \dots, y_n^*]^\top) &= \frac{\text{Cov}^2([x_1|\dots|x_n]^\top w, [y_1, \dots, y_l, y_{l+1}^*, \dots, y_n^*]^\top)}{\sigma^2([x_1|\dots|x_n]^\top w)\sigma^2([y_1, \dots, y_l, y_{l+1}^*, \dots, y_n^*]^\top)} \approx \\ \frac{\text{Cov}^2([x_1|\dots|x_l]^\top w, [y_1, \dots, y_l]^\top)}{\sigma^2([x_1|\dots|x_n]^\top w)} &= \dots = \frac{w^\top Nw}{w^\top Cw}, \quad N = (\sum_i^l y_i x_i)(\sum_i^l y_i x_i)^\top, \quad C = \sum_i^n x_i x_i^\top \end{aligned}$$

Die Etiketten y_{l+1}^*, \dots, y_n^* , kennen wir nicht, deswegen approximieren wir im zweiten Schritt die Kovarianz nur anhand der etikettierten Daten. Es sei hier auch bemerkt, dass wir angenommen haben, dass $\sum_i^l x_i \approx 0$, was für ausreichend viel Graphen aus $\sum_i^n x_i = 0$ folgt. Ähnlich folgt aus $\sum_i^l y_i^2 = 1$, dass $\sum_i^n y_i^2 \approx 1$ und aus $\sum_i^l y_i = 0$, dass $\sum_i^n y_i \approx 0$. Deswegen konnten wir annehmen, dass $\sigma^2([y_1, \dots, y_l, y_{l+1}^*, \dots, y_n^*]^\top) \approx 1$. Nachdem wir das Eigenwert-Problem gelöst haben, ist der Wert des i -ten generierten Merkmals für einen Graphen x durch $x^\top w_i$ gegeben. Wichtig ist, dass die Summe in der Berechnung von C über alle Graphen geht, nicht nur über die etikettierten. Dies entspricht der Tatsache, dass man zwar die Etiketten zwar für die Berechnung der Kovarianz braucht, aber nicht der Standardabweichung. Es ist klar, dass die Matrix C dann positiv definit ist, wenn es mindestens m linear

²In der Praxis wird bei der eigentlichen Implementierung oft der umgekehrte Ansatz verfolgt: Man löst das Eigenwertproblem indem man in einem Iterationsverfahren die Eigenwerte mit dem Rayleigh-Koeffizienten approximiert. Für uns ist das unwesentlich.

³Wenn die Matrix C zusätzlich nichtsingulär ist, wird das Problem einfacher – siehe [52], Seite 132.

unabhängige Graphenvektoren x_i gibt. Wenn das nicht zutrifft ist sie immer noch positiv semidefinit. Es ist nicht schwierig zu sehen, dass hier die Nullraum-Annahme erfüllt ist⁴

Das Verfahren kann auch auf Kernels verallgemeinert werden [65]. Es kann überprüft werden, dass die obige Version dem linearen Kernel entspricht. Wir verzichten⁵ nun auf die Annahme $\sum_i x_i = 0$ und gehen nur von der Normierung der Etiketten aus: $\sum_i y_i = 0$ und $\sum_i y_i^2 = 1$. Gegeben sei ein positiv definites Kernel K , die entsprechende Funktion $\Phi : \mathbb{R}^m \rightarrow \mathcal{H}$ und das Skalarprodukt \cdot (Erläuterungen siehe Abschnitt zu SVMs: 2.3.1). Bezeichnen wir nun $w = \sum_i \alpha_i \Phi(x_i)$. Das neue Merkmal ist durch $[w \cdot \Phi(x_1), \dots, w \cdot \Phi(x_n)]^\top$ definiert. Schreiben wir jetzt die Kovarianzformel.

$$\begin{aligned} \text{Cor}^2([w \cdot \Phi(x_1), \dots, w \cdot \Phi(x_n)]^\top, [y_1, \dots, y_l, y_{l+1}^*, \dots, y_n^*]^\top) &= \\ \frac{\text{Cov}^2([w \cdot \Phi(x_1), \dots, w \cdot \Phi(x_n)]^\top, [y_1, \dots, y_l, y_{l+1}^*, \dots, y_n^*]^\top)}{\sigma^2([w \cdot \Phi(x_1), \dots, w \cdot \Phi(x_n)]^\top) \sigma^2([y_1, \dots, y_l, y_{l+1}^*, \dots, y_n^*]^\top)} &\approx \\ \frac{\text{Cov}^2([w \cdot \Phi(x_1), \dots, w \cdot \Phi(x_l)]^\top, [y_1, \dots, y_l]^\top)}{\sigma^2([w \cdot \Phi(x_1), \dots, w \cdot \Phi(x_n)]^\top)} & \end{aligned}$$

Wir führen nun ein paar Notationen ein. Der Kernel K wird als Matrix betrachtet, und zwar so, dass die etikettierten Graphen vor den unetikettierten kommen. Die Matrix K besteht also aus den folgenden Teilen.

$$K = \left[\begin{array}{c|c} K_L & B^\top \\ \hline B & \end{array} \right], \quad K' = [K_L | B^\top], \quad K' : l \times n, \quad K : n \times n, \quad K_L : l \times l$$

Die Kovarianz kann nun wie folgt berechnet werden.

$$\begin{aligned} \text{Cov}^2([w \cdot \Phi(x_1), \dots, w \cdot \Phi(x_l)]^\top, [y_1, \dots, y_l]^\top) &= \left(\sum_i^l (w \cdot \Phi(x_i)) - \sum_j (w \cdot \Phi(x_j)) y_i \right)^2 = \\ \left(\sum_i^l \sum_j^l \alpha_j (\Phi(x_j) \cdot \Phi(x_i)) y_i - (\sum_j^l (w \cdot \Phi(x_j))) (\sum_i^l y_i) \right)^2 &= \\ \left(\sum_i^l \sum_j^l \alpha_j (\Phi(x_j) \cdot \Phi(x_i)) y_i \right)^2 &= \\ (\alpha^\top K_L [y_1, \dots, y_l]^\top)^2 &= \alpha^\top (K_L [y_1, \dots, y_l]^\top) (K_L [y_1, \dots, y_l]^\top)^\top \alpha \end{aligned}$$

Damit wir weiterrechnen können, brauchen wir noch die Matrix F zu definieren (die Notation $A_{.,i}$ bedeutet die i -te Spalte der Matrix A).

$$F_{.,i} = \sum_{k \neq i}^n K'_{.,k} \quad F : l \times n$$

Jetzt berechnen wir die Standardabweichung.

$$\begin{aligned} \sigma^2([w \cdot \Phi(x_1), \dots, w \cdot \Phi(x_n)]^\top) &= \sum_i^n ((w \cdot \Phi(x_i)) - \sum_k^n (w \cdot \Phi(x_k)))^2 = \\ \sum_i^n (\sum_{k \neq i}^n (w \cdot \Phi(x_k)))^2 &= \sum_i^n (\sum_{k \neq i}^n \sum_j^l \alpha_j (\Phi(x_j) \cdot \Phi(x_k)))^2 = \sum_i^n (\sum_{k \neq i}^n \sum_j^l \alpha_j K_{jk})^2 = \\ \sum_i^n (\sum_{k \neq i}^n \alpha^\top K'_{.,k})^2 &= \sum_i^n (\alpha^\top F_i)^2 = \alpha^\top F F^\top \alpha \end{aligned}$$

⁴Überhaupt hat hier die Matrix N nur einen nichttrivialen Eigenvektor, es kann hier also nur ein Merkmal generiert werden.

⁵In [65] wird fälschlicherweise behauptet, dass $\sum_i^n x_i = 0$ eine Normierung im Hilbertraum impliziert: $\sum_i^n \Phi(x_i) = 0$. Da sich dies in den Herleitung niederschlägt, wird sie jetzt ganz wiederholt.

Nun kann der Rayleigh-Koeffizient geschrieben werden.

$$\text{Cor}^2([w \cdot \Phi(x_1), \dots, w \cdot \Phi(x_n)]^\top, [y_1, \dots, y_l, y_{l+1}^*, \dots, y_n^*]^\top) = \frac{\alpha^\top (K_L[y_1, \dots, y_l]^\top)(K_L[y_1, \dots, y_l]^\top)^\top \alpha}{\alpha^\top F F^\top \alpha}$$

Es sei hier bemerkt, dass wir hier nicht mehr direkt nach w optimieren, sondern nach den Koeffizienten α . Nachdem wir α berechnet haben, kann für den Graphen x_q das k -te generierte Merkmal mit der Formel $\sum_i^l \alpha_i^{(k)} K_{iq}$ berechnet werden. Es bleibt zu begründen, warum die Matrix $F F^\top$ positiv definit ist. Jede Matrix der Form $A A^\top$ ist positiv semidefinit. Es reicht also zu argumentieren, warum F nichtsingulär ist. Für jede Matrix mit Spalten $[a_1, \dots, a_n]$ gilt aber, dass a_i und a_j genau dann linear unabhängig sind, wenn $\sum_{k \neq i} a_k$ und $\sum_{k \neq j} a_k$ linear unabhängig sind. Daraus und aus der Nichtsingularität von K_L folgt die positive Definitheit von F . Wenn der Kernel singulär (also nur positiv semidefinit) wäre, müsste man ein anderes als das oben beschriebene Verfahren konstruieren, da hier die Nullraum-Annahme nicht unbedingt erfüllt ist. Im Kontext dieser Arbeit, wo man gewöhnlich mehr Untergraphen hat als Moleküle, ist das aber eher unwahrscheinlich.

1.2.4 Hauptkomponentenanalyse

Ein noch weiterer Ansatz zur Merkmalsuche kommt aus der Statistik. Hauptkomponentenanalyse unterscheidet sich darin von den zuvor besprochenen Ansätzen, dass man nicht aus den vorhandenen Merkmalen diejenigen betrachten soll, die irgendwelches Kriterium erfüllt, wie wir zuvor angenommen haben, sondern dass man diese Menge sozusagen komprimiert — es entsteht am Ende ein anderer Satz von Merkmalen, der zwar auf dem ursprünglichen basiert, doch keine Untermenge von ihm ist [74]. Damit wir das Verfahren definieren können, müssen wir zuerst die Merkmalmatrix definieren.

Definition 10. Die MERKMALMATRIX ist eine Matrix mit Elementen aus \mathbb{R} , die als $A_{ij} = 1$ wenn $b_{s_j} \subseteq b_{g_j}$ und ansonsten als $A_{ij} = 0$ definiert ist.

Wir führen nun die Matrix A_{norm} , die einer Normierte Version von A ist. Wir bezeichnen die Zeilen dieser Matrix als $GV_{norm}(\mathbf{g}_i)$. Es gilt jetzt wegen der Normierung, dass $\sum_{i=1}^{i=|\mathcal{G}|} el(i, GV_{norm}(\mathbf{s}_k)) = 0$. Wir suchen also eine Matrix P , so dass $P A_{norm} (P A_{norm})^T$ eine diagonale Matrix D ist. Dazu kann die SVD-Matrixzerlegung [78] benutzt werden. Es kann einfach überprüft werden, dass die Matrizen $P = U^\top$, $D = \Sigma^2$ unsere Bedingung erfüllen.

$$U^\top A_{norm} (U^\top A_{norm})^\top = U^\top U \Sigma V^\top (U^\top U \Sigma V^\top)^\top = \Sigma V^\top (\Sigma V^\top)^\top = \Sigma V^\top V \Sigma = \Sigma^2$$

HKA ist von der Definition aus für unetikettierte Beispiele sehr gut geeignet, da die Etiketten für die Berechnung überhaupt nicht wichtig sind. Eine Möglichkeit, einige Beschränktheiten dieses Ansatzes zu vermeiden wäre es, die Definition des Skalarproduktes zu ändern und Matrizen über einem anderen Körper als \mathbb{R} zu betrachten (Kernel-HKA). HKA kann auch als ein besondere Fall des Rayleigh-Koeffizienten behandelt werden. Wegen Zeitmangels wurde in dieser Arbeit HKA nicht untersucht.

1.2.5 Vergleich der Ansätze

In der nachfolgenden Tabelle 1.2 werden die hier angesprochenen Ansätze für Merkmalsuche bei Regression miteinander verglichen. Alleine von der theoretischen Seite her überzeugt die Idee mit dem Rayleigh-Koeffizienten. Wie wir im Abschnitt 3.6 sehen werden, konnte dies in den praktischen Experimenten leider nicht bestätigt werden.

Tabelle 1.2 Vergleich verschiedener Merkmalsuchverfahren.

	Pearson	MI	Rayleigh	HKA
Teilüberwachtes Lernen	✗	✗	✓	✓
Korr. untereinander	✗	✗	✓	✓
Korr. mit Target	✓	✓	✓	✗
Zeitbedarf (theoretisch $O(\cdot)$) ^a	kn	knd	$k^2n + k^3$	$k^2n + k^3$

^a Abkürzungen: k – Anzahl der Merkmale, n – Anzahl der Graphen. Bei MI bedeutet $d = d(n)$ die Komplexität der Funktion, die eindimensionale Dichte approximiert (bei Parzen-Fenstern $n=$). Die Angabe zu HKA bezieht sich auf die Situation, wo man SVD berechnet. Es sind schnellere Iterationsverfahren möglich, wurden hier aber nicht beschrieben. Die Angabe zum Rayleigh Koeffizienten basiert auf der kubischen Komplexität des allgemeinen QZ-Algorithmus. Es sind auch hier je nach Kernel Optimierungen möglich.

1.3 Ausblick: Konstruktion von Merkmalen

In dieser Arbeit wurde (außer bei der Diskussion von HKA und dem Rayleigh-Koeffizienten) stets angenommen, dass man die einmal extrahierten Merkmale nicht modifizieren, sondern nur eine Untermenge davon auswählen kann. Man muss sich aber nicht darauf beschränken: Es gibt eine Reihe von Ansätzen, in denen versucht wird, entweder existierende Merkmale miteinander zu verbinden, oder von Anfang an gezielt generische Merkmale zu konstruieren. Srinivasn und King [77] kodieren Moleküle mithilfe von Prolog-Prädikaten, woraufhin Prolog-Regeln generiert werden, die die eingespeisten Informationen zusammenfassen. Anhand dieser Regeln können dann neue, generische Merkmale generiert werden. Borgelt et.al. [12] stellen ein Verfahren zur Extraktion von Untergraphen vor, die Wildcard-Atome beinhalten (die aber von dem Benutzer vorgegeben werden müssen, das heißt die Merkmalsextraktion ist nicht automatisch). Der Ansatz kann auch erweitert werden, so dass ähnliche Untergraphen zu solchen mit OR-Knoten automatisch verbunden werden [56], wobei das χ^2 -Maß eines solchen generischen Moleküls als Stoppkriterium für das Miteinbeziehen neuer Untergraphen benutzt wird. Es ist schließlich auch möglich andere Varianten des Rayleigh-Koeffizienten zu betrachten [58]. Dann können bekannte Invarianzen und Informationen zum Rauschen benutzt werden. Dies wurde nach dem Wissen des Autors noch nicht gezielt für Untergraphen gemacht.

1.4 Anmerkungen zur Graphenextraktion

Der Fokus bei dieser Arbeit liegt auf Merkmalsuche und Generalisierung, den Techniken der Extraktion wird nur ein kurzer Überblick gewidmet. Wir fangen damit an, das Verfahren formal zu definieren.

Definition 11. Ein UNTERGRAPHENEXTRAKTIONSVERFAHREN ist ein Algorithmus, der einen Satz von Graphen $\mathcal{G} = \mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_n$ als Eingabe entgegennimmt und die Menge von Untergraphen $\mathcal{S} = \mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_k$ sowie eine Relation \subseteq zurückliefert, wobei $\forall \mathbf{s} \in \mathcal{S} \exists \mathbf{g} \in \mathcal{G}. \mathbf{s} \subseteq \mathbf{g}$ und die Relation \subseteq gleich mit der üblichen Untergraphenrelation ist, die auf die Menge $\mathcal{S} \times \mathcal{G}$ projiziert wurde. Ferner kann das Suchverfahren optional die besagte Relation auch für die Menge $\mathcal{S} \times \mathcal{S}$ zurückliefern. Die optionale Bedingung dient dazu, dass das Merkmalsuchverfahren (und eventuell der Klassifizierer) auf die Gitterstruktur der extrahierten Merkmale zugreifen kann. Bei allen dem Autor bekannten Ansätze erhöht sie nicht den algorithmischen Aufwand der Extraktion.

Tabelle 1.3 Zugehörigkeit des Graphenisomorphismus (G-ISO), Untergraphenisomorphismus (U-ISO) und des Problems des maximalen gemeinsamen Untergraphen (MGU) zu Klassen P und NP.

	G-ISO	U-ISO / MGU
beliebige Graphen	NP	NP-vollständig ^a
beschr. Baumweite	P [55]	P / NP-vollständig ^b [55]
beschr. Grad	P [54]	NP-vollständig [36]
planar	P [39]	NP-vollständig ^c [36]

^a polynomiell, wenn der Untergraph-Kandidat fixiert ist.

^b P wenn man zusätzlich annimmt, dass der Untergraph-Kandidat beschränkten Grad hat [55].

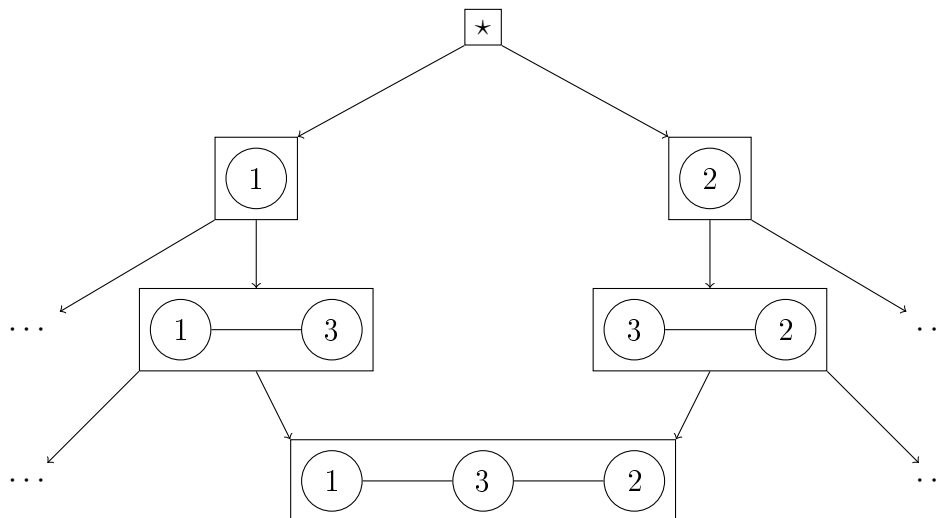
^c Linear, wenn der Untergraph-Kandidat fixiert ist [31].

Bevor wir praktische Ansätze zur Extraktion von Graphen vorstellen, werden wir uns kurz mit der theoretischen Seite des Problems befassen. Betrachten wir zuerst drei folgende Entscheidungsverfahren: Gegeben seien zwei Graphen, sind sie isomorph? (Graphenisomorphismus). Gegeben seien zwei Graphen, ist einer davon isomorph mit einem Untergraphen des anderen? (Untergraphenisomorphismus). Gegeben seien drei Graphen, ist der dritte Graph ein maximaler gemeinsamer zusammenhängender Untergraph der zwei ersten? (maximaler gemeinsamer Untergraph). Stellen wir zuerst einmal fest, dass die so definierten Probleme des Untergraphenisomorphismus und des maximalen gemeinsamen Untergraphen aus der Sicht der Zugehörigkeit zur P oder NP gleichwertig sind. Wir werden jetzt die bestehenden Ergebnisse zur Zugehörigkeit der obigen Probleme zu P und NP vorstellen (siehe Tabelle 1.3). Die Tabelle sagt grundsätzlich zwei Dinge aus: Erstens ist Graphenisomorphismus höchstwahrscheinlich einfacher zu lösen, als Untergraphenisomorphismus, zweitens muss man etwas über die Eigenschaften der Graphen annehmen, damit effiziente Algorithmen möglich werden.

Moderne Ansätze zur Extraktion von Untergraphen basieren auf DFS-Suche des Untergraphengitters [91] (siehe Grafik 2, man kann dies als eine Version des Eclat-Algorithmus [97], der für allgemeine Objektsätze konzipiert ist, verstehen). Wenn wir von der DFS-Suche ausgehen, gibt es von der theoretischen Seite her anhand der Tabelle 1.3 zwei Möglichkeiten effizient vorzugehen: Entweder speichern wir alle Einbettungen, in welchem Falle wir nur Isomorphismus testen müssen, oder wir speichern sie nicht, in welchem Falle wir das schwierigere Problem des Untergraphenisomorphismus lösen müssen. Welcher Ansatz davon der richtige ist, kann man im allgemeinen nicht sagen, da es von dem Datensatz abhängt: Wenn der Datensatz groß ist, aber nur wenig Muster extrahiert werden, lohnt es sich, Einbettungen zu speichern, ansonsten kann es sein, dass der Speicherbedarf zu hoch wird. Ebenfalls hängt vieles von den Eigenschaften der untersuchten Graphen aus. Für chemische Molekülgraphen kann man zum Beispiel folgendes sagen.

1. Ausnahmslos alle Molekülgraphen haben beschränkte Valenz.
2. Die meisten Molekülgraphen, die man untersuchen möchte, sind planar [67]. Gegenbeispiele muss man meistens gezielt konstruieren, klassische Beispiele nichtplanarer Moleküle befinden sich z.B. bei Sauvage [72].
3. Die meisten Molekülgraphen haben Baumbreite, die kleiner gleich drei ist. Eine Studie [92] hat ergeben, dass es in der LIGAND-Datenbank nur ein Molekül gibt, das Baumweite vier hat (dies entspricht 0.01% des Datensatzes) und keins mit größerer Baumweite.

Abbildung 2 Beispiel eines Untergraphengitters bei Extraktion von Graphen. Die Ziffern stehen für Etiketten der Knoten. Der Stern steht für den leeren Graphen, der Untergraph jedes Graphen ist.



Eine weitere Studie [40] zeigt, dass es in dem NCI-Datensatz nur 65 Moleküle mit Baumweite, die größer ist als vier gibt (dies entspricht 0.03% des Datensatzes).

Diese Eigenschaften deuten gemeinsam mit der Tabelle 1.3 auf zwei theoretisch fundierte Lösungswege hin: Wenn wir nun ein effizientes Verfahren zur Extraktion von Untergraphen konstruieren wollen, haben wir die folgenden zwei Möglichkeiten: Wenn das Speichern von Einbettungen möglich ist, kann man die beschränkte Valenz (oder die nahezu-Planarität) ausnutzen, und Isomorphismus schnell testen; wenn dies nicht in Frage kommt, kann man sich auf der Baumweite-Schranke gemeinsam mit beschränkter Valenz stützen, so dass sogar der Untergraphenisomorphismus polynomiell lösbar wird. Es sei hier noch bemerkt, dass ein solcher Baumweite-basierter Algorithmus die Graphen mit zu hoher Baumweite effizient identifizieren kann: Obwohl die Berechnung der Baumweite NP-vollständig ist [2], kann sie für jede im voraus bekannte Baumweite polynomiell berechnet werden (dies folgt zum Beispiel aus dem Robertson-Seymour-Theorem). Als Fazit der theoretischen Überlegungen könnte man sagen: Die Theorie der NP-Vollständigkeit kann nicht dazu benutzt werden, Graphenextraktion aus chemischen Datenbanken, als schwierig zu bezeichnen.

Nichtsdestotrotz sind nach dem Wissen des Autors alle bisherigen praktischen Ansätze heuristisch⁶. Bei der Konstruktion einer heuristisch motivierten DFS-Graphenextraktion müssen zwei grundlegende Probleme gelöst werden [91]: Erstens müssen die Fragmente effizient (mit möglichst wenig Duplikaten) enumeriert werden — Duplikate verändern zwar nicht das Endergebnis, verschlechtern aber die Performanz, zweitens muss bei der Erweiterung der Fragmente effizient feststellbar sein, in welchen Molekülen sich das gerade erweiterte Fragment befindet.

In dieser Arbeit wurde der MoFa-Miner [12] benutzt. Bei dem MoFa-Miner wird effiziente Enumerierung dadurch erreicht, dass Atome und Bindungen Indizes bekommen, die dann benutzt werden, um lokale Codes zu generieren, die dann gewisse (nicht alle) Arten von Iso-

⁶Die Autoren des einzigen Algorithmus [40], der mit der Theorie der NP-Vollständigkeit (und dem Begriff der Baumweite) motiviert ist, haben ihn nicht implementiert.

morphismen ausschalten. Das zweite Problem wird durch das Speichern aller Einbettungen gelöst. Ein anderer Ansatz, der für diese Arbeit ausprobiert aber am Ende nicht benutzt wurde, ist gSpan [93], wo eine kanonische Form der Graphen benutzt wird (minimaler DFS-Code⁷), um isomorphe Fragmente auszuschalten, bei Erweiterung der Fragmente sind explizite Tests für Untergraphenisomorphismus nötig.

Eine gewünschte Eigenschaft von Graphenextraktionsverfahren ist die Fähigkeit, nur geschlossene Untergraphen zurückzuliefern. Bei der DFS-Suche kann dies besonders einfach implementiert werden, indem man die Erweiterungen der Fragmente, die die Anzahl und die Verteilung der Einbettungen in den einzelnen Molekülen nicht verändern (die sogenannten perfekten Erweiterungen), vor allen anderen Erweiterungen untersucht — was den Suchbaum deutlich verkleinert. Die Frage ist, wie die Verteilungen der Einbettungen effizient verglichen werden können: in MoFa [57] kann das direkt gemacht werden, da MoFa die Einbettungen speichert. In gSpan sind spezielle Überlegungen nötig, die aber effizient implementiert werden können (siehe [94] für Details).

Worlein et. al. [91] zeigen eindeutig, dass MoFa der langsamste der vier getesteten Ansätze ist (die anderen drei waren gSpan, FFMS und Gaston). Die Entscheidung, bei der Vorbereitung dieser Arbeit fiel trotzdem auf MoFa — und das aus praktischen Gründen, da dies die einzige dem Autor bekannte frei verfügbare und stabile Software ist, die das Mining von geschlossenen Fragmenten unterstützt und neben der Graphenstruktur der Fragmente auch die Information zur Untergraphenrelation bzgl. der ursprünglichen Moleküle zurückliefert. Es war nötig, nur geschlossene Untergraphen zu betrachten, da deren Anzahl ansonsten zu hoch wurde. Für die hier betrachteten Datensätze war MoFa bei der Extraktion geschlossener Fragmente schneller als gSpan bei der Extraktion aller Fragmente⁸.

⁷Die Berechnung des minimalen DFS-Codes ist exponentiell lang, für die praktischen Anwendungen in der Chemie jedoch schnell genug.

⁸Es gibt keine vom Autor frei erhältliche Implementierung von gSpan mit der Extraktion geschlossener Fragmente. In MoFa ist eine unabhängige Implementierung eingebaut, die aber nicht getestet wurde, da die betrachteten Datensätze klein genug sind, um das Speichern der Einbettungen zu ermöglichen.

Kapitel 2

Klassifizierungs- und Regressionsverfahren

In diesem Kapitel werden die Verfahren vorgestellt, die nach der Merkmalsuche angewandt wurden, um die Etiketten vorauszusagen. Es wird jeweils untersucht, inwiefern das Verfahren unetikettierte Daten berücksichtigt und ob es die Möglichkeit gibt, von einer transduktiven Aufgabenstellung zu profitieren.

2.1 Nächster Nachbar

Ein sehr häufiger Ansatz bei Objekten, die keinem wohlbekanntem Raum angehören, sind die k -nächster-Nachbar-Methoden (kNN), da man nur die Metrik benötigt, und die Annahme dass die Wahrscheinlichkeitsverteilung der Objekte kontinuierlich ist¹. Es ist auch sofort klar und auch formell einfach ausdrückbar ist, was der induktive Bias ist². Der Algorithmus sieht in seiner einfachsten Variante so aus, dass man zuerst anhand der Trainingsmenge eine Datenbank konstruiert. Bei der anschließenden Anfrage wird das Objekt in der Datenbank gesucht, das bezüglich der gewählten Metrik dem zu Klassifizierenden Objekt am nächsten ist und sein Etikett wird zurückliefert [30], [26]. Es kann bewiesen werden [30], dass das Verfahren stochastisch konvergiert, wenn die Größe der Datenbank $n \rightarrow \infty$. Zum Verhältnis zu Bayes-Optimalklassifizierer gibt es das folgende Erkenntnis.

Theorem. *Die Wahrscheinlichkeit eines Fehlers im 1-NN-Verfahren P erfüllt die Formel $P_B \leq P \leq 2P_B$, wobei P_B die Quote für den Bayesschen Optimalklassifizierer ist.*

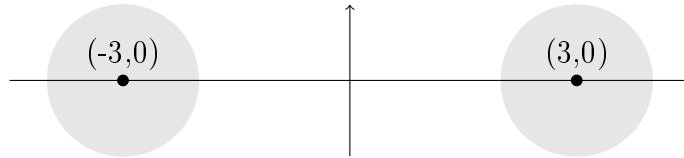
Der Beweis [30] wird hier nicht angegeben. Das Theorem betrifft nur den Fall der Konvergenz im Limit, sagen aber nicht dazu aus, wie schnell er erreicht wird. Es gibt dazu negative Ergebnisse, die Konvergenz kann beliebig langsam sein (sieh etwa das Beispiel von Cover [25]). In der Praxis wird oft mehr Nachbarn als einer in Betracht gezogen. Dies verbessert nicht immer das Konvergenzverhalten des Algorithmus (siehe 3). Die Frage der Komplexität ist asymptotisch gesehen uninteressant, da der naive Algorithmus die Komplexität $O(n)$ hat und sie auch offensichtlich optimal ist. Nichtsdestotrotz gibt es Ansätze [30], [41], die die Beschleunigung des Verfahrens zum Ziel haben.

Wenn mehrere Nachbarn in Betracht gezogen werden, ist es oft zweckmäßig, die Etiketten mit den Abständen zu gewichten. das heißt im Falle der Regression nimmt man den gewichteten Durchschnitt und bei einer endlichen Menge von Etiketten nimmt man

¹Die letztere Bedingung kann man auflockern, was aber den Konvergenzbeweis erschwert.

²Dies bedeutet einen Vorteil gegenüber z.B. Entscheidungsbäumen, wo man außer der allgemeinen Bemerkung, dass der Algorithmus kleinere Bäume bevorzugt zum induktiven Bias wenig sagen kann.

Abbildung 3 Verteilung, für die 1-NN besser ist als kNN, $k > 1$. Der Radius beider Kreise beträgt 1. Die Verteilung innerhalb der Kreise ist uniform, außerhalb der Kreise hat jeder Wert Wahrscheinlichkeit null. Punkte in beiden Kreisen haben unterschiedliche Etiketten. Quelle: [26].



für jedes mögliche Etikett c eines Nachbarn \mathbf{x}' in der betrachteten Nachbarschaft N den Wert $\sum_{\mathbf{x}' \in N, L(\mathbf{x}')=c} 1/(1 + d(\mathbf{x}, \mathbf{x}'))$ und nimmt dann das Etikett mit dem größten Wert. Auch bei der Berechnung der Metrik $d(\cdot, \cdot)$ selbst kann man die unterschiedlichen Merkmale unterschiedlich gewichten. In dieser Arbeit wurde der gewichtete Tanimoto-Abstand als Metrik benutzt. Als Gewichte der Merkmale W_s wurden im kategorischen Fall die χ^2 -Werte genommen, bei Regression wurde das Quadrat der Pearson-Korrelation zwischen dem Merkmal und der Zielvariable benutzt (siehe Abschnitt 1.2.1). Die Varianz und Kovarianz werden dann anhand der verfügbaren Daten approximiert. Zum Vergleich wurde auch der im Weka [38] Verfügbare Standard-kNN-Klassifizierer³ herangezogen. Er betrachtet statt der Metrik Werte der Ähnlichkeit und liefert ein Etikett zurück, für das der Wert $-\sum_{\mathbf{x}' \in N, L(\mathbf{x}')=c} \sqrt{\sum_i |el(MV^{\mathbb{R}}(\mathbf{x}), i) - el(MV^{\mathbb{R}}(\mathbf{x}'), i)|}$ maximiert wird [1]. Hier werden die unterschiedlichen Korrelationen der Merkmale mit der Zielvariable ignoriert.

2.2 Entscheidungsbäume und J48graft

Ein weit verbreiteter Ansatz zur Klassifizierung von Daten mit Attributen, die entweder kategorisch sind, oder die zwar numerisch sind, aber sich mit Weniger-als-Vergleichen gut behandeln lassen, sind Entscheidungsbäume. Er scheint auch für unseren Fall von binären Attribute gut geeignet. Da es sich bei dem Begriff „Entscheidungsbaum“ um eine ganze Familie von Algorithmen handelt, deren vollständige Beschreibung den Rahmen dieser Arbeit sprengen würde, beschränken wir uns jetzt auf die Behandlung einer Version davon⁴, die in Weka [38] implementiert wurde, und auf C4.5 [48] basiert.

Der Algorithmus [48] funktioniert in drei Phasen: Zuerst wird ein Baum konstruiert, dessen Knoten den möglichen Werten von Attributen entsprechen und dessen Knoten Untermengen der Trainingsmenge entsprechen; dann werden in einem Pruning-Verfahren Teile davon entfernt. Anschließend werden in einem Grafting-Verfahren Teile hinzugefügt. Die Konstruktion erfolgt auf der Greedy-Basis: Man durchsucht den Raum von Bäumen indem man jeweils das Attribut wählt, so dass die Teilung des aktuellen Knoten auf der Basis dieses Attributs den größtmöglichen Informationsgewinn aufweist. Der Informationsgewinn ist ein Begriff aus der Informationstheorie und ist wie folgt definiert.

$$G = I(\mathbf{x}) - \sum_{a \in \{0,1\}} \frac{|Graphen(\mathbf{x} \rightsquigarrow \mathbf{s}_a)|}{|Graphen(\mathbf{x})|} I(\mathbf{x} \rightsquigarrow \mathbf{s}_a)$$

³weka.classifiers.lazy.IBk

⁴weka.classifiers.trees.J48graft

$$\text{wobei } I(\mathbf{k}) = - \sum_{c \in V} f(c, \mathbf{k}) \log f(c, \mathbf{k}) \quad \text{und} \quad f(c, \mathbf{k}) = \frac{|\{\mathbf{g} \in \text{Graphen}(\mathbf{k}) : l(\mathbf{g}) = c\}|}{|\text{Graphen}(\mathbf{k})|}$$

Die Summierung in der Berechnung von G erfolgt über die Möglichen Werte der Attribute (in unserem Fall gib es nur zwei da ein Untergraph entweder da ist oder nicht). Mit \mathbf{x} wird der aktuelle Knoten im Baum gemeint, mit $\mathbf{x} \rightsquigarrow \mathbf{s}_a$ ein hypothetischer Nachfolger, der um das Attribut \mathbf{s} erweitert wurde. Die Menge $\text{Graphen}(\mathbf{k})$ steht hier für die Menge von Graphen, die dem Knoten \mathbf{k} korrespondiert. Das Verfahren wird so lange durchgeführt, bis es keine Attribute mehr gibt, die einen Gewinn an Information bewirken: Nach dem Durchlauf der ersten Phase des Algorithmus sind in jedem Blatt des Baumes also zwei Situationen möglich: Entweder gibt es dort nur Objekte mit einem Etikett oder aber es gibt Objekte mit mehreren Etiketten, die anhand der verfügbaren Attribute nicht voneinander unterschieden werden können (Inkonsistenzen in der Trainingsmenge). Um die Verallgemeinerungsfähigkeit des Klassifizierers zu steigern, wird danach Pruning und Grafting durchgeführt. Die Verfahren werden hier nicht in allen Einzelheiten beschrieben, zwei Möglichkeiten, Pruning durchzuführen sind in der Abbildung 4 zu sehen. Grafting wird auf der Basis der benachbarten Objektmenge (Abbildung 5) durchgeführt, die Idee ist der Grafik 6 zu entnehmen.

2.3 SVMs, TSVMs und Varianten davon

Support Vector Machines sind ein universeller Ansatz, Klassifizierer zu konstruieren, deren Grundlagen in den 60er Jahren von Vladimir Vapnik entwickelt wurden und der ab den 90ern immer häufiger zum praktischen Einsatz kommt. Sie sind, neben neuronalen Netzen, einer der zwei universellen Architekturen, die weniger eine konkrete Instanz eines Klassifizierers darstellen, sondern die Art und Weise angeben, wie einer konstruiert werden kann. Sie basieren auf der Idee, dass man beim Trainieren eines Klassifizierers nicht nur die Fehlerquote in Betracht ziehen soll, die man mit den eingegebenen Trainingsdaten erzielt, sondern auch den „Rand“, der die Klassen trennt. Im linearen Fall bedeutet dies, dass man nach einer Hyperfläche sucht, die zwei Klassen von Objekten voneinander trennt. Der „Rand“ entspricht dann dem Abstand zwischen der Fläche und einer Parallelen Fläche, die das am nächsten gelegene Objekt beinhaltet. Die Tatsache, dass man im Hinblick auf zwei Ziele optimiert, kann man auch als eine Variante von Regularisierung sehen. Im nichtlinearen Fall (das heißt wenn man die kanonische Definition des Skalarproduktes durch eine eigene ersetzt) gibt es keine intuitive geometrische Intuition, man kann aber immer noch von „Support Vectors“ reden, also von den Objekten, die auf dem „Rand“ liegen. Deswegen heißt der Ansatz auch Support Vector Machine.

2.3.1 Überwachtes Lernen

Wir werden den Ansatz jetzt für den überwachten Fall formal definieren [18]. Gegeben sei eine Funktion $\Phi : \mathbb{R}^d \rightarrow \mathcal{H}$, wobei \mathcal{H} ein Hilbertraum mit dem Skalarprodukt \cdot ist. Eine Support-Vector-Machine für binäre Klassifizierung ist ein Verfahren, das den Wert des folgenden Ausdrucks minimiert.

$$\frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i, \quad w \in \mathcal{H}, \quad C \in \mathbb{R}, \quad \xi_i \in \mathbb{R}$$

Dabei sind die Nebenbedingungen wie folgt.

$$\forall i. \xi_i \geq 0, \quad \forall i. y_i(w \cdot \Phi(\mathbf{x}_i) + b) \geq 1 - \xi_i$$

Abbildung 4 Zwei Möglichkeiten, wie im C4.5 Algorithmus Pruning durchgeführt werden kann. Das Bild (a) zeigt den ursprünglichen Baum. Der Unterbaum k kann entweder unverändert bleiben, durch ein Blatt ersetzt werden (b), oder sein Nachfolger k' kann an die Stelle von k „hochgezogen“ werden.

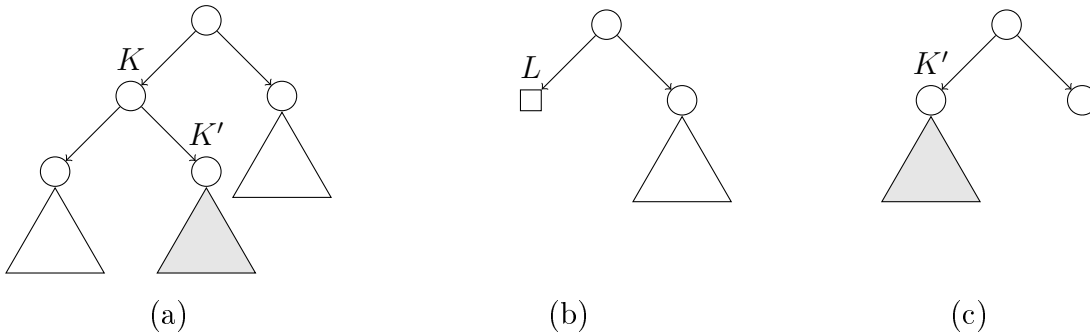


Abbildung 5 Der Bereich der $nb(l)$ Trainingsmenge, den das Blatt l benachbart ist hellgrau schattiert. Der Bereich $obj(l)$ ist dunkel schattiert.

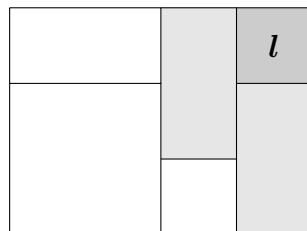
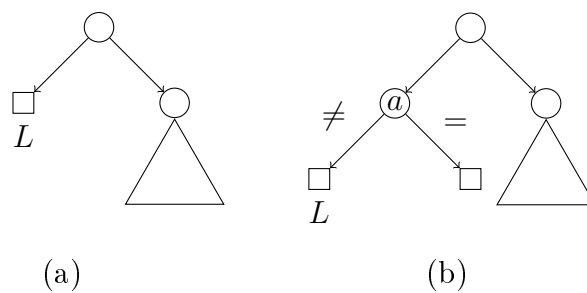


Abbildung 6 Ein Entscheidungsbaum vor (a) und nach (b) Grafting.



Die Werte $y_i \in \{-1, 1\}$ stehen hier für die Etikettierung der jeweiligen Knoten \mathbf{x}_i . Der Parameter C , der vor dem Start des Trainings bestimmt wird, dient dazu, Probleme zu behandeln, wo die zwei Klassen nicht voneinander sauber getrennt werden können: Je größer der Wert, desto höher die „Strafe“ für jeden Punkt, der in der falschen Klasse landet, wobei diese „Strafe“ von der Entfernung bezüglich des gewählten Skalarproduktes abhängt. Man kann auch sagen, dass der Wert dieses Parameters unser Vertrauen in die Kombination von Trainingsdaten und in die Auswahl des induktiven Bias, also des Raumes \mathcal{H} und der Transformation Φ , ausdrückt. Bei guter Modellierung, also in einer Situation, wo \mathcal{H} und Φ auf die Zielvariable gut abgestimmt sind, erwarten wir wenig falsch klassifizierte Trainingsdaten und nehmen stattdessen einen schmalen Rand in Kauf (der Ausdruck $\|w\|$ entspricht der Breite des Randes — so wird bei SVMs das Prinzip der Regularisierung implementiert).

Damit SVMs praktisch anwendbar sind, gilt es, eine explizite Berechnung von Φ und \cdot zu vermeiden. Dies ist deshalb wichtig, weil man für komplizierte Probleme einen möglichst „reichen“ Raum \mathcal{H} verwenden möchte (z.B. einen hochdimensionalen), in dem eine Direkte Berechnung von \cdot von dem Zeitbedarf her teuer ist. Die Regel, wie eine fertig trainierte Maschine benutzt werden kann, um Aussagen über neue Objekte zu treffen, kann wie folgt formuliert werden.

$$f(x) = \Phi(\mathbf{x}) \cdot w + b = \sum_{s \in \text{Sup}V} \alpha_s y_s \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}_s) + b \quad (*)$$

Die Menge $\text{Sup}V$ in den oben angegebenen Gleichungen beinhaltet die Indices von Support-Vektoren. Sie ist als $\text{Sup}V = \{i : \alpha_i > 0\}$ definiert. Die entscheidende Beobachtung, die man anhand der obigen Formeln machen kann, ist dass sowohl die zu optimierende Funktion in der Wolfe-Form (siehe [18]) als auch die Gleichung (*) nur von den Ausdrücken der Form $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}')$ abhängig ist, wobei x und x' die Trainingsobjekte und in der späteren Auswertung die zu klassifizierenden Objekte sind. Aufgrund dieser Beobachtung kann man darauf verzichten, Φ und \mathcal{H} explizit zu definieren. Stattdessen definiert man lediglich eine Funktion K (den Kernel), die direkt das Skalarprodukt \cdot in \mathcal{H} berechnet, ohne eine Konversion durchzuführen: $K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}')$. Der zusätzliche konzeptuelle Gewinn, den man durch die Anwendung von Kernels erzielt, besteht neben der Beschleunigung der Berechnung darin, dass man K auch als eine $n \times n$ -Matrix interpretieren kann, die im Voraus berechnet wird⁵.

Die Frage, wie die Optimierung praktisch implementiert werden ist kompliziert und sprengt die Rahmen dieser Arbeit. Nach Burges [18] können wir aber sagen, dass es Algorithmen gibt, die im Fall, wo die Anzahl der Support-Vektoren $|\text{Sup}V|$ klein ist im Vergleich zu der Anzahl n der Objekte überhaupt die Komplexität $O(|\text{Sup}V|^3 + |\text{Sup}V|^2 n + |\text{Sup}V| n d_L)$ aufweisen, wobei d_L die Anzahl der Attribute (das heißt die Dimension des ursprünglichen Raumes) ist.

Was noch bleibt, ist die Frage, wie K zu wählen ist. Hierfür gibt es folgende Überlegungen: Erstens kann man den Raum \mathcal{H} und die Funktion Φ explizit definieren (wobei man dann natürlich Beweisen muss, dass der Raum ein Hilbertraum ist). Dann setzt man $K(\mathbf{x}, \mathbf{x}')$ explizit als $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}')$. Die Konstruktion eines solchen Raumes kann aber schwierig sein. Deswegen kann man auch eine a-priori Definition für K konstruieren, und beweisen, dass es ein \mathcal{H} und Φ gibt, so dass die $K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}')$ gilt. Ein Werkzeug dafür liefert das Mercersche Theorem, das besagt, dass sich $K(\mathbf{x}, \mathbf{x}')$ genau dann als eine Summe $\sum_i \Phi(\mathbf{x})_i \Phi(\mathbf{x}')_i$ ausdrücken lässt, wenn für alle Funktionen g , $\int g^2 < \infty$ gilt, dass $\int K(\mathbf{x}, \mathbf{y}) g(\mathbf{x}) g(\mathbf{y}) d\mathbf{x} d\mathbf{y} \geq 0$. Dieses Theorem gibt also eine ausreichende und notwendige Bedingung dafür, dass ein Ker-

⁵Es kann sein, dass eine solche Berechnung im voraus den gesamten Algorithmus langsamer macht. Es kann auch sein (insbesondere wenn der Raum \mathcal{H} ein euklidischer Raum mit kleiner Dimension ist), dass es schneller ist, direkt zu optimieren und auf Kernels und die Wolfe-Form ganz zu verzichten.

nel einem Hilbertraum entspricht, der auf dem Punktprodukt basiert⁶. Da das Training des Algorithmus die Werte des Kernels nur für eine (endliche) Menge der Trainingsobjekte benutzt, ist damit auch ein ad-hoc-Mechanismus verwandt, mit dem man feststellen kann, ob der Algorithmus für eine gewisse Funktion, die als Kernel eingegeben wird aber deren Kerneleigenschaft nicht bewiesen ist, terminieren wird: Es reicht zu überprüfen ob $K(\mathbf{x}_1, \dots, \mathbf{x}_n)$, die wir nun als quadratische Matrix betrachten, positiv semidefinit ist.

2.3.2 Teilüberwachtes Lernen

Jetzt können wir dazu übergehen, wie die unetikettierten Daten miteinbezogen werden können. Definieren wir zuerst das entsprechende Optimierungsproblem. Nehmen wir an, dass die unetikettierten Objekte Indices von $l + 1$ bis n haben.

$$\frac{1}{2} \|w\|^2 + C \sum_{i=1}^l \xi_i + C^* \sum_{i=l+1}^n \xi_i, \quad w \in \mathcal{H}, \quad C, C^* \in \mathbb{R}, \quad \xi_i \in \mathbb{R}$$

Dabei sind die Nebenbedingungen wie folgt.

$$\forall i. \xi_i \geq 0, \quad \forall i. y_i(w \cdot \Phi(\mathbf{x}_i) + b) \geq 1 - \xi_i,$$

Dabei ist zu Bemerkem, dass die y_i für $i > l$ unbekannt sind und im Optimierungsverfahren neben w , b und ξ_i erst berechnet werden müssen. Die letztere Nebenbedingung kann auch umgeformt werden, so dass die unbekanntem Werte dort nicht vorkommen.

$$\begin{aligned} \forall i. y_i(w \cdot \Phi(\mathbf{x}_i) + b) \geq 1 - \xi_i &\Leftrightarrow (*) \\ \forall i \leq l. y_i(w \cdot \Phi(\mathbf{x}_i) + b) \geq 1 - \xi_i \quad \wedge \quad \forall i > l. |w \cdot \Phi(\mathbf{x}_i) + b| \geq 1 - \xi_i \end{aligned}$$

Welche der zwei Formen bevorzugt werden soll hängt mit der Auswahl des Lösungsansatzes zusammen, wir werden diese Frage noch später besprechen.

Damit die Verlustfunktion, die durch das obige Optimierungsproblem implizit definiert wird explizit ausgedrückt werden kann, werden wir das Optimierungsproblem noch einmal umformen, und die Hilfsvariablen ξ eliminieren.

$$\frac{1}{2} \|w\|^2 + C \sum_{i=1}^l L(y_i, w \cdot \Phi(\mathbf{x}_i) + b) + C^* \sum_{i=l+1}^n L(y_i, w \cdot \Phi(\mathbf{x}_i) + b)$$

Die Verlustfunktion kann nun als $L(y, o) = \max(0, 1 - yo)$ definiert werden (über y_i , $i > l$ wird optimiert). Analog zu (*) können wir die Funktion auch ohne diese unbekanntem werte von y_i ausdrücken, und nämlich $L_U(o) = \max(0, 1 - |o|)^7$ [19]. Die zwei Funktionen L , L_U sind in der Abbildung 7 zu sehen. Die Funktion L ist (für ein gegebenes y) konvex, die Funktion L_U ist es nicht. Man hat bei der Lösung also die Wahl zwischen nichtkonvexer Optimierung einerseits, und konvexer Optimierung, wobei zusätzlich die richtigen Werte von y_i , $i > l$ ermittelt werden müssen, andererseits.

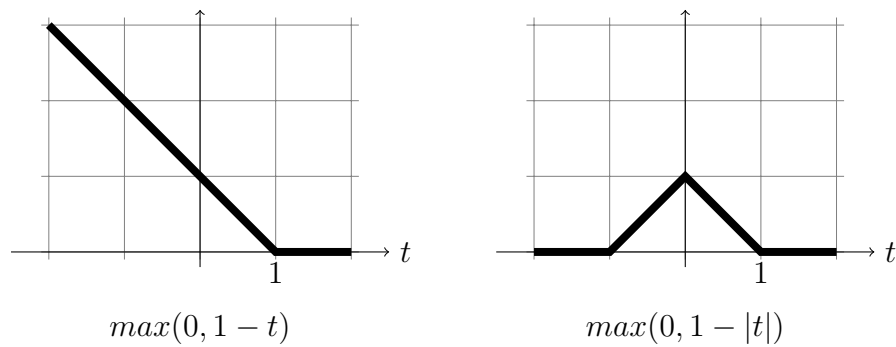
Die Wolfe-Form ist identisch mit der für den überwachten Fall, mit der Ausnahme, dass die Nebenbedingungen jetzt wie folgt sind.

$$\forall i \leq l. 0 \leq \alpha_i \leq C, \quad \forall i > l. 0 \leq \alpha_i \leq C^*, \quad \forall i. \sum_i^n \alpha_i y_i = 0$$

⁶Es gibt, Hilberträume, wo das Skalarprodukt anders definiert ist als durch das Punktprodukt (und wo Elemente keine mit einer natürlichen Zahl indizierbaren Vektoren sind). Die Familie der Räume, wo man nur das Punktprodukt betrachtet ist aber für die meisten Anwendungen reich genug.

⁷Die Funktion L_U kann auch als Realisierung des Clustering-Ansatzes gesehen werden, da sie die Situation bestraft, in der Punkte verschiedener Klassen nah aneinander sind.

Abbildung 7 Konvexe und nichtkonvexe Verlustfunktionen bei (T)SVMs.



Bei der Lösung des Optimierungsproblems in Wolfe-Form sind y_i unbekannt, das heißt man sucht solche y_i , dass der maximale Wert der Wolfe-Formel möglichst klein ist.

$$\min_{y_{l+1}, \dots, y_n} \max_{\alpha_1, \dots, \alpha_n} W(y_{l+1}, \dots, y_n, \alpha_1, \dots, \alpha_n)$$

Für den teilüberwachten Fall trifft alles zu, was wir über die Berechnung des Skalarproduktes \cdot und über Kernels in dem vorigen Abschnitt gesagt haben.

Damit die errechnete Lösung „balanciert“ ist, muss noch eine zusätzliche Nebenbedingung eingeführt werden [21], die aussagt, dass die Zuweisungen der Labels zu den unbekanntem y_i , so sein müssen, dass sich daraus ein bestimmtes Verhältnis r zwischen den Labels ergibt: $(n - l)^{-1} \sum_{l+1}^n \max(y_i, 0) = r$. Bei bestimmten Verfahren, wo eine Implementierung der obigen Bedingung schwierig wäre, können andere (im allgemeinen Fall nicht äquivalente) Balanzierconstraints eingeführt werden.

Wir werden uns jetzt mit der Frage befassen, wie das Optimierungsproblem gelöst werden kann. Erstens kann man die Probleme, die durch die Nichtkonvexität der zu optimierenden Funktion entstehen, dadurch vermeiden, dass man mehrere Kombinationen für die Werte von y_i ausprobiert und anschließend beste Lösung auswählt, also eine übliche SVM 2^n -Mal trainiert. Man kann die Laufzeiten mit dem Branch-and-Bound-Ansatz verbessern, wo die Knoten des Entscheidungsbaumes den verschiedenen möglichen Zuweisungen von Werten für y_i entsprechen. Den Wert einer SVM, die mit Objekten mit bereits bekannten Etiketten trainiert wurde, nimmt man dann als untere Schranke [20] für den Wert der Optimierungsfunktion⁸. Eine andere Möglichkeit ist es, das Problem als Ganzzahlprogrammierung⁹ [7] zu betrachten. Es sind keine Algorithmen bekannt, die die optimale Lösung liefern und schnell genug sind, um mit größeren Datensätzen zu arbeiten. Deswegen bestehen alle bisherigen Ansätze, die sich skalieren lassen, auf approximierten Lösungen des nichtkonvexen Optimierungsproblems. Hier werden sie nicht angesprochen, eine ausführliche Auslegung befindet sich in [21], siehe auch einen Frühen Ansatz von Joachims [42]. Natürlich gibt bei keinem dieser Verfahren eine formale Garantie dafür, dass das Endergebnis ein globales Minimum ist. In der Tat zeigen die Untersuchungen in [21], dass die approximierten Ansätze die global optimale Lösung manchmal weit verfehlen.

⁸Man kann es machen, weil $C, C^*, L \geq 0$.

⁹Dieses ist dann aber möglicherweise schwieriger zu lösen, als das Ursprungsproblem.

2.3.3 Regression

Wir werden jetzt das Optimierungsproblem für den Fall formulieren, wo die Zielvariable eine kontinuierliche Größe ist (eine reelle Zahl). Die Grundidee dabei ist es, dass man nur die Abweichungen der Zielfunktion von ihrer Approximation bestraft, wo der Wert der Abweichung mehr als eine vorgegebene Größe ϵ beträgt. Das ist ein Unterscheidungsmerkmal von den klassischen Methoden der Funktionsapproximation (z.B. Regression mit Polynomen), wo man zumeist von einer Quadratischen Verlustfunktion ausgeht, die auch für kleine Abweichungen ungleich null ist. Formulieren wir nun das Optimierungsproblem [75].

$$\frac{1}{2}\|w\| + C \sum_{i=1}^n \xi_i + \xi_i^*, \quad w \in \mathcal{H}, \quad C \in \mathbb{R}, \quad \xi_i, \xi_i^* \in \mathbb{R}$$

Dabei sind die Nebenbedingungen wie folgt.

$$\forall i. \xi_i, \xi_i^* \geq 0, \quad \forall i. y_i - w \cdot \Phi(\mathbf{x}_i) - b \leq \epsilon + \xi_i, \quad \forall i. w \cdot \Phi(\mathbf{x}_i) + b - y_i \leq \epsilon + \xi_i^*$$

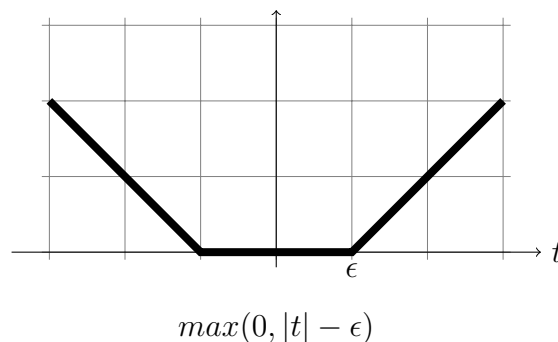
$$b \in \mathbb{R}, \quad \epsilon, y_i \in \mathbb{R} \text{ (vorgegeben)}$$

Wir werden das Problem jetzt umformen, so dass die benutzte Verlustfunktion explizit erkennbar wird.

$$\frac{1}{2}\|w\| + C \sum_{i=1}^l L_{Reg}(y_i, w \cdot \Phi(\mathbf{x}_i) + b)$$

Die Funktion $L_{Reg}(y, o) = \max(0, |y - o| - \epsilon)$ ist in der Abbildung 8 zu sehen; sie spiegelt den oben angesprochenen Ansatz wider, dass kleine ($\leq \epsilon$) Abweichungen zulässig sind. Es kann einfach überprüft werden, dass die Funktion konvex ist, was bedeutet, dass Regression ähnlich schnell durchgeführt werden kann, wie Klassifizierung.

Abbildung 8 Verlustfunktion bei Regression mit SVMs.



Die durch die Anwendung des Algorithmus berechnete Funktion ist durch die Gleichung $f(x) = w \cdot \Phi(\mathbf{x}) + b$ gegeben und entspricht einer Hyperfläche im \mathcal{H} . Für lineare Kernels reduziert dies auf eine euklidische Fläche, wo ein Punkt durch die gleiche Anzahl der Parameter beschrieben werden kann, wie die Anzahl der Elemente (Merkmale) in x . Wenn der Vektor x nur Länge 1 hat, ist die berechnete Funktion also eine übliche lineare Funktion in \mathbb{R} .

Die Wolfe-Form wird hier nicht angegeben (siehe [75]). Wir beschließen die Behandlung der Regression lediglich mit der Bemerkung, dass auch hier, wie im Fall der Klassifizierung, nur das Wissen über den Kernel benötigt wird.

2.3.4 Die Unmöglichkeit transduktiver Regression mit SVMs

Wenn man die Regression- und Transduktionsansätze, die in den vorigen Abschnitten besprochen wurden betrachtet, liegt sofort die Frage nahe, ob man diese nicht vereinigen könnte und eine transduktive SVM für Regression bauen könnte. Die Antwort darauf ist negativ. Wie wir bald sehen werden, würde das zu einer trivialen Lösung führen, die unbrauchbar ist. Dies bedeutet natürlich nicht, dass die Idee der SVM sich nicht auf irgendeine Art und Weise in einen Algorithmus für transduktive Regression einbinden lässt, sondern lediglich, dass diese Einbindung keine natürliche ist und zusätzliche Schritte notwendig macht. Wir werden uns mit entsprechenden Verfahren im Abschnitt 2.4 befassen. Betrachten wir nun eine naive Übertragung des TSVM-Ansatzes auf Regression. Lösen wir das folgende Optimierungsproblem [23].

$$\min_{w, b, y_{l+1}, \dots, y_n} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l L_{Reg}(y_i, w \cdot \Phi(\mathbf{x}_i) + b) + C^* \sum_{i=l+1}^n L_{Reg}(y_i, w \cdot \Phi(\mathbf{x}_i) + b)$$
$$w \in \mathcal{H}, \quad C, C^* \in \mathbb{R}, \quad w \in \mathbb{R}$$

Nehmen wir nun an, dass w, b die Lösung des überwachten Teils des Problems ist (also $C^* = 0$). Dann sieht man Folgendes: Da vernünftige Verlustfunktionen nichtnegativ sind, kann das Setzen von C^* auf eine positive Zahl den Wert der Optimierungsfunktion nur vergrößern. Dann sind die „optimalen“ Werte von $y_i, i > l$ aber genau gleich $f(\mathbf{x}_i)$, da für eine solche Zuweisung dieser optimale Wert der Funktion erreicht wird. Da die Lösung die gleiche ist, die wir im überwachten Fall bekommen werden die unetikettierten Daten nicht richtig ausgenutzt.

2.4 Transduktive Regression

Wie wir im vorigen Abschnitt gesehen haben, ist es nicht trivial, Verfahren, die für überwachtetes Lernen konzipiert wurden, auf teilüberwachtes Lernen zu übertragen. Überhaupt ist es bisher nicht gelungen, ein Verfahren für transduktive Regression zu konstruieren, das, wie die SVMs im überwachten Fall, breite theoretische Fundierung hat und in einer Vielzahl von Situationen anwendbar ist. Stattdessen gibt es mehrere Ansätze, die zwar eine (Teil-)Begründung in der Theorie haben, doch nicht eindeutig als der jeweils beste betrachtet werden können. Wir werden nun einige davon besprechen.

2.4.1 Zweischrittverfahren

Man kann ein Verfahren der transduktiven Regression als zweiteilig betrachten: Einerseits ist die Ausnutzung lokaler Informationen nötig, um die Kontinuität¹⁰ der Lösung sicherzustellen, andererseits reicht sie nicht aus, um das volle Potenzial der Daten auszuschöpfen — es ist nötig, global zu optimieren. Ein natürlicher Ansatz, dies zu implementieren, ist es, ein zweistufiges Verfahren zu konstruieren [23]. Nehmen wir an, wir haben eine Kernelfunktion, die die Anforderungen eines SVM-Kernels erfüllt. Dann können wir in der ersten Phase die Etiketten der unetikettierten Objekte ermitteln, indem wir den NN-Ansatz anwenden, wobei wir entweder die Kernelfunktion als Ähnlichkeitsmaß betrachten, oder eine andere Metrik auf \mathcal{H} definieren. Dabei werden nur Punkte benutzt, die in einem bezüglich dieser Metrik definierten Abstand von dem gerade betrachteten Punkt liegen. Nun, da alle y_i bekannt sind,

¹⁰Sie kann je nach Ansatz anders verstanden und definiert werden.

kann man das folgende Optimierungsproblem lösen [23].

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l L_{Reg}(y_i, w \cdot \Phi(\mathbf{x}_i) + b) + C^* \sum_{i=l+1}^n L_{Reg}(y_i, w \cdot \Phi(\mathbf{x}_i) + b)$$

$$w \in \mathcal{H}, \quad C, C^* \in \mathbb{R}, \quad w \in \mathbb{R}$$

Dies ist eine übliche Regression-SVM - der einzige Unterschied besteht darin, dass wir zwei konstanten C, C^* haben, was aber an der Implementierung wenig ändert¹¹. Üblicherweise möchte man natürlich $C^* < C$ setzen, da man weniger Vertrauen in die ermittelten als in die vorgegebenen Werte hat. Nachdem man den SVM-Schritt durchgeführt hat, werden nicht mehr die von der NN-Regel generierten Werte von y_i benutzt, sondern diejenigen, die sich aus der SVM-Lösung ergeben.

2.4.2 Direkte Regularisierung

Ein anderer Ansatz ist es, die SVMs ganz zu vergessen und stattdessen ein ganz neues Optimierungsproblem zu formulieren. Betrachten wir das folgende Problem [24].

$$\min_f (f - y)^T C (f - y) + f^T R f, \quad f, y \in \mathbb{R}^n, \quad R, C \in \mathbb{R}^{n \times n}, \quad R, C \text{ symmetrisch} \quad (*)$$

Der wesentliche Unterschied zum vorigen Ansatz besteht darin, dass der gesuchte Vektor f nicht die Parameter einer Funktionsfamilie darstellt, sondern die zu approximierenden Werte der Funktion selbst (das Verfahren ist rein transduktiv, es wird keine Entscheidungsregel generiert). Der Vektor y beinhaltet für $i \leq l$ die bekannten Werte der zu approximierenden Funktion y_i , für die unetikettierten Punkte setzt man meistens $y_i = 0$ ¹². Dieses ist eine sinnvolle Annahme, wenn die Werte der Anwendung des Algorithmus normiert wurden, und wenn wir einen geeigneten Regularisierungsansatz benutzen, der die Kontinuität der Ausgabe erzwingt. Die Optimierungsformel kann wie folgt interpretiert werden: Der Ausdruck $(f - y)^T C (f - y)$ stellt sicher, dass die ermittelten Werte von f nah sind an die bekannten Werte y . Der Ausdruck $f^T R f$ stellt sicher, dass die berechneten Werte f für benachbarte Argumente nicht zu weit auseinandergehen (Regularisierung). Wir werden jetzt eine einfache Art und Weise darstellen, wie die Matrizen R, C definiert werden können. Setzen wir zuerst C auf $\text{diag}(C_L, \dots, C_L, C_{UL}, \dots, C_{UL})$, wobei die C_{UL} den unetikettierten Punkten entsprechen. Jetzt definieren wir die Matrix R . Betrachten wir zuerst den gewichteten NN-Graphen für die gegebene Objektmenge. Dieser ist ein vollständiger Graph mit n Knoten und bezüglich der benutzten Metrik d (man kann natürlich auch hier Kernels anwenden) wie folgt definiert¹³ [82].

$$w_{ij} = \exp\left(\frac{-d(\mathbf{x}_i, \mathbf{x}_j)}{\sigma}\right)$$

Die Intuition hinter der Benutzung der Exponentialfunktion und nicht direkt einer einfacheren Ähnlichkeitsfunktion, die durch die Metrik d induziert ist, besteht in der Analogie zum Irrfahrten-Prozess. Wenn wir cw_{ij} als Wahrscheinlichkeit des Überganges von i zu j (c ist eine normierende Konstante) interpretieren, und die Übergänge unabhängig sind, beträgt die Wahrscheinlichkeit, dass wir von i nach j über k in zwei Schritten kommen, $c^2 w_{ik} w_{kj}$. Diese

¹¹In der dualen Form ändern sich nur die Schranken für $\alpha_i, \alpha_i^*, i > l$, ansonsten können gleiche Lösungsansätze benutzt werden, wie für übliche Regression.

¹²Man kann sich natürlich auch vorstellen, dass sie aus einem Schritt der lokalen Approximation stammen, wie im vorigen Ansatz. Diese Idee wurde hier nicht verfolgt.

¹³Wenn die Metrik auf einem Punktprodukt basiert, kann man den Koeffizienten σ auch von dem Index abhängig machen, also mehrere σ_i betrachten, die man dann das Zielproblem anpasst [98].

Multiplizierung der Gewichte entspricht der Summierung der Werte von d , die im Exponent sind — das ist gut, da die natürliche Arbeitsweise mit den Werten einer Metrik Summierung und nicht Multiplizierung ist. Bezeichnen wir nun die Matrix mit den Elementen w_{ij} als W und die diagonale Matrix $\text{diag}(\sum_j^n w_{1,j}, \dots, \sum_j^n w_{n,j})$ als D . Die Matrix R definieren wir wie folgt.

$$R(W) = D(W) - W$$

Dies ist die Standarddefinition der Laplace-Matrix eines Graphen. Für jede Laplacematrix, die auf einer Gewichtung W basiert kann die quadratische Form $f^\top R f$ als $\sum_{ij} w_{ij} (f_i - f_j)^2$ geschrieben werden [76]. Dies entspricht genau der Eigenschaft, die wir von der Regressionslösung wollen: kleine Abweichungen für Objekte, die nah aneinander sind, also für die w_{ij} groß ist. Statt der hier angegebenen Laplacematrix kann man auch eine normierte Variante davon benutzen [90], dieser Ansatz wird hier nicht verfolgt. Wenn wir die Matrizen R, C berechnet haben, bleibt es nur, eine analytische Lösung des Optimierungsproblems (*) auszurechnen. Sie ist durch die folgende Formel gegeben [24].

$$f_{opt} = (R + C)^{-1} C y$$

Der hier definierte Regressionsalgorithmus hat also Komplexität $O(n^3)$, da wir eine Matrix der Größe $n \times n$ invertieren müssen¹⁴.

Es sind natürlich auch andere Definitionen von R, C möglich, eine nähere Besprechung befindet sich in [24]. Man kann sogar in der Matrix R die Bedingung kodieren, dass die durch den Regressionsalgorithmus ermittelten Werte die folgende Bedingung erfüllen: Der Wert jedes Punktes soll möglichst nah dem Wert stehen, den eine lokal trainierte SVM zurückliefert (Details siehe [90]).

2.4.3 Regularisierung in einer Funktionsfamilie

In dem vorgängigen Abschnitt haben wir ein Verfahren definiert, das direkt die gesuchten Werte der Funktion in bestimmten Punkten ermittelt, ohne dass eine Approximation der Funktion auf ganzer Domäne stattfindet. Dies hat den Nachteil, dass man ohne ein Modell der zu approximierenden Funktion einzuführen auch keine Glattheitseigenschaften dieser Funktion (bezüglich der ganzen Domäne) in dem Optimierungsproblem spezifizieren kann. Wir werden dies jetzt tun, indem wir eine explizit definierte Funktionsfamilie einführen. Dadurch werden sowohl Glattheitsbedingungen auf der ganzen Domäne möglich als auch jene, die in dem durch die bekannten (auch unetikettierten) Punkte definierten Bereich der erhöhten Dichte (die wir schon im vorigen Abschnitt behandelt haben). Diese entsprechen zweier Arten von Regularisierung. Definieren wir nun das Optimierungsproblem [4].

$$\begin{aligned} & \min_f (f_x - y)^\top C (f_x - y) + \gamma_I f_x^\top R f_x + \gamma_A \|f\|_{\mathcal{H}_K} \\ & y \in \mathbb{R}^n, \quad f \in \mathcal{H}_K, \quad f_x = [f(\mathbf{x}_1), \dots, f(\mathbf{x}_n)]^\top, \quad \gamma_I, \gamma_A \in \mathbb{R} \\ & R = D - W \text{ wie in 2.4.2,} \quad C = \text{diag}(\overbrace{C_1, \dots, C_l}^{1..l}, \overbrace{0, \dots, 0}^{l+1, \dots, n}) \end{aligned}$$

\mathcal{H}_K ist dabei ein Hilbertraum mit reproduzierendem Kernel K (wir nehmen hier an, dass K die Anforderungen eines Kernels erfüllt, die im Abschnitt 2.3.1 erläutert wurden). Bezeichnen wir die Anzahl der Merkmale als d . \mathcal{H}_K besteht aus Funktionen, die lineare Kombinationen von Funktionen vom Typ $\lambda \mathbf{x} \cdot K(\mathbf{x}, \mathbf{x}') : \mathbb{R}^d \rightarrow \mathbb{R}$ sind, wobei jedem $\mathbf{x}' \in \mathbb{R}^d$ eine solche

¹⁴Man könnte natürlich W so definieren, dass sie dünn besetzt ist (z.B. Objekte abseits einer Sphäre gar nicht berücksichtigen. Dieser Ansatz wird hier nicht verfolgt.)

Funktion entspricht¹⁵. Nehmen wir an, dass $f, g \in \mathcal{H}_K$ in der gleichen abzählbaren Basis ausgedrückt sind, so dass \mathcal{B} die entsprechende Objektmenge ist.¹⁶: $f = \sum_{\mathbf{b} \in \mathcal{B}} \alpha_{\mathbf{b}}(\lambda \mathbf{x} \cdot K(\mathbf{x}, \mathbf{b}))$, $g = \sum_{\mathbf{b} \in \mathcal{B}} \beta_{\mathbf{b}}(\lambda \mathbf{x} \cdot K(\mathbf{x}, \mathbf{b}))$. Dann ist das Skalarprodukt $f \cdot g : \mathcal{H}_K \times \mathcal{H}_K \rightarrow \mathbb{R}_+ \cup \{0\}$ als $\sum_{(\mathbf{b}, \mathbf{b}') \in \mathcal{B} \times \mathcal{B}} \alpha_{\mathbf{b}} \beta_{\mathbf{b}'} K(\mathbf{b}, \mathbf{b}')$ definiert. Die Norm $\|f\|_{\mathcal{H}_K}$ ist die kanonische Norm in \mathcal{H}_K : $\|f\|_{\mathcal{H}_K} = \sqrt{f \cdot f}$ [43]. Eine (um $f_x^\top R f_x$) erweiterte Form des Repräsentationstheorems [4] zeigt, dass die Funktion f_{opt} , die das Optimierungsproblem minimiert, eine lineare Kombination von Komponenten $\lambda \mathbf{x} \cdot K(\mathbf{x}, \mathbf{x}_i), i = 1, \dots, n$ ist. Insbesondere bedeutet dies, dass die Basis des Raumes der zu betrachteten Funktionen endlich ist. Bezeichnen wir nun als α_i die Koeffizienten von f_{opt} in dieser Basis.

$$f_{opt}(x) = \sum_{i=1}^n \alpha_i K(\mathbf{x}, \mathbf{x}_i)$$

Es kann bewiesen werden [4], dass diese Koeffizienten mit der folgenden Formel analytisch berechnet werden können.

$$\alpha = (JK + \gamma_A C_l^{-1} I + \gamma_I C_l^{-1} R K)^{-1} y$$

Diesen Ansatz kann man natürlich verallgemeinern. Man könnte die in dem Optimierungsproblem vorkommenden Matrizen anders definieren oder aber eine ganz andere Verlustfunktion einführen, als solche, die sich als eine quadratische Form $(f_x - y)^\top M (f_x - y)$ schreiben lässt. In [4] z.B. wird das Verfahren auch für die SVM-Verlustfunktion behandelt.

2.4.4 Metrikbasierte Ansätze

Eine andere Möglichkeit, Regularisierung durchzuführen, besteht in der folgenden Beobachtung. Betrachten wir unseren Hypothesenraum und eine Metrik, die es uns ermöglicht, Funktionen in diesem Raum zu vergleichen [73]. Da wir nur über eine endliche Anzahl von Trainingspunkten und nicht über explizite Definitionen der Funktionen verfügen, können wir Abstände in dem Raum nur Approximiert messen, und zwar mit zwei verschiedenen Granularitäten. Erstens können wir den Abstand $d(f, h)$ zwischen der Zielfunktion f und einer Hypothese h nur ungenau messen, da wir wenig etikettierte Objekte haben. Wir bezeichnen diese Messung als d_L , da zur Berechnung die Etiketten erforderlich sind. Den Abstand $d(h_1, h_2)$ zwischen zwei Hypothesen können wir dagegen genauer berechnen, da wir beide Funktionen evaluieren können und nur unetikettierte Daten brauchen. Bezeichnen wir diese Messung als $d_{L,UL}$. Von einer Hypothese, die Funktion f möglichst gut approximiert erwarten wir, dass der *wahre* Abstand $d(f, h)$ klein ist. Betrachten wir nun das folgende (empirische) Optimierungsproblem [73].

$$\min_h d_L(f, h) \max \left(\frac{d_L(h_0, h)}{d_{L,UL}(h_0, h)}, \frac{d_{L,UL}(h_0, h)}{d_L(h_0, h)} \right)$$

Der Faktor $d_L(h_0, h)$ misst die Eignung der Hypothese, der andere Faktor misst mithilfe einer vorgegebenen Hypothese h_0 die Qualität dieser Messung. Meistens nimmt man zu diesem Zweck den Durchschnittswert der Zielvariable als eine konstante Funktion. Man kann ihn auch als eine Form der Regularisierung verstehen. Der Vorteil dieses Ansatzes ist es, dass es einfach ist und nach [73] gut funktioniert. Der Nachteil ist, dass man eine besondere Hypothese h_0 angeben muss, die sich auf komplizierte Art und Weise auf das Endergebnis auswirkt. Außerdem kann es je nach Definition der Abstände schwierig sein, eine geschlossene Formel für h zu errechnen. In dieser Arbeit wurde dieser Ansatz nicht praktisch untersucht.

¹⁵Es kann natürlich sein, dass zwei verschiedene Vektoren in \mathbb{R}^d der gleichen Funktion entsprechen.

¹⁶Wir nehmen an, dass eine existiert.

2.4.5 Vergleich zwischen den Ansätzen

In diesem Abschnitt werden die drei Regressionsansätze miteinander verglichen (siehe Tabelle 2.1). Man sieht sofort, dass Regularisierung in einer Funktionsfamilie eigentlich von der theoretischen Seite her der Beste Ansatz ist. Dagegen sprechen aber folgende Überlegungen. Erstens ist direkte Regularisierung dann besser, wenn wir nun an reiner Transduktion interessiert sind, und keine globale Regularisierung wollen, da wir uns nicht mit dem relativ komplizierten Apparat von Funktionsräumen beschäftigen müssen. Zweitens kann es sein, dass Zweischrittverfahren insbesondere für sehr große Datensätze doch interessant werden, weil sehr viel Forschungsarbeit in die algorithmische Verbesserung von SVM-Trainingsverfahren investiert worden ist. Es gibt auch bereits eine Mehrzahl von fertigen Implementierungen davon, die heftig optimiert worden sind.

Tabelle 2.1 Vergleich verschiedener Verfahren zur teilüberwachter Regression (ZV - Zweischrittverfahren, DR - direkte Regularisierung, RF - Regularisierung in einer Funktionsfamilie).

	ZV	DR	RF
globale Regularisierung	✓	✗	✓
lokale Regularisierung	✓	✓	✓
theoretisch fundiert	✗	✓	✓
globale Entscheidungsregel	✓	✗	✓
Komplexität	$O(SV ^3)^a$	$O(n^3)^b$	$O(n^3)^b$

^a Viele Optimierungen möglich (z.B. dünnbesetzte Kernels, lineare Kernels etc.).

^b Optimierbar (z.B. wenn man andere Verlustfunktion nimmt), Approximationsansätze möglich.

Kapitel 3

Pipeline und Auswertung

In diesem Kapitel wird zuerst die benutzte Pipeline vorgestellt, dann werden die erzielten Ergebnisse besprochen.

3.1 Methoden der Auswertung

In diesem Abschnitt werden die methodologischen Grundlagen der Auswertung beschrieben. Da die meist verwandten Techniken der Auswertung von Klassifizierern, wie Kreuzvalidierung, Holdout usw. nur Heuristiken¹ sind, werden wir jetzt versuchen ihre starken und schwachen Seiten zu behandeln.

3.1.1 Die Heuristiken im überwachten Fall

Eine der am meisten angewandten Heuristiken, die bei überwachtem Lernen Anwendung finden, ist k -fache Kreuzvalidierung. Eine Basismethode [46], mit der die Heuristiken oft verglichen werden, ist Holdout — man nimmt einfach einen Teil der Daten weg und testet mit dem Rest. Ein häufiger Vorwurf gegen Holdout ist, dass die Daten ineffizient benutzt werden. Dies kann gemindert werden, indem man die Aufteilung des Datensatzes in zwei Mengen zufällig macht und das ganze iteriert. Es kann unter bestimmten Annahmen formell gezeigt werden, dass dies tatsächlich zu besseren Ergebnissen führt². k -Fache Kreuzvalidierung fußt auf der Idee, dass die meisten Klassifizierer stabil sind, das heißt sich für eine geringfügig verkleinerte Trainingsmenge ähnlich verhalten wie für eine größere. Nach [46] ist dies in der Wirklichkeit von realen Klassifizierern nur dann erfüllt, wenn sie nah an ihrer Lernkapazität sind; es können Beispiele angegeben werden [46], wo die Stabilität nicht gegeben ist. Es lässt sich noch die folgende erstaunliche Bemerkung machen: Wenn es tatsächlich so wäre, dass ein Klassifizierer völlig stabil, also für Kreuzvalidierung geeignet, wäre, dann würde die Varianz des ermittelten Performanzwertes nicht von der Anzahl der Folds k abhängen. Es gilt allgemein, dass zeitintensive Verfahren nicht unbedingt zur Verbesserung der Qualität

¹ In [89] wird bewiesen, dass Verfahren zur Beurteilung von Klassifizierern Meta-Klassifizierer sind und ebenso wie die üblichen Klassifizierer einen induktiven Bias brauchen. In [6] wird mittels statistischer Werkzeuge gezeigt, dass sich diese Varianz der Trefferquote eines Klassifizierungsverfahrens überhaupt nicht auf eine theoretisch begründbare Weise genau ermitteln lässt. Dies hat natürlich schwere Folgen, wenn man Klassifizierer miteinander vergleichen möchte, da es für die in einem solchen Fall typischerweise angewandten Methoden der einfachen Statistik (Annahme der Normalverteilung mit einem gewissen Mittel und Standardabweichung) gar keine Begründung gibt.

²Die Annahmen, die in [9] gemacht werden, entsprechen nicht ganz der Anwendung in der Praxis, da man davon ausgeht, dass die Endhypothese durch das Auswählen eines zufälligen Folds entsteht und nicht durch das Lernen mit der ganzen Trainingsmenge. Inwiefern man mit dieser Annahme einverstanden ist, hängt von der Einstellung ab, die man zum Problem der Stabilität des jeweiligen Klassifizierers hat.

der geschätzten Größe führen [46]. Ein weiteres Problem, das sowohl Kreuzvalidierung als auch Holdout aufweisen, ist, dass die zwei Mengen, mit denen man testet und trainiert, nicht wirklich unabhängig sind. Zum Beispiel ist bei zwei-Klassen-Problemen die Anzahl der Objekte einer Klasse in der Testmenge durch die Anzahl der Objekte derselben Klasse in der Trainingsmenge eindeutig determiniert. Man kann versuchen, diesem Phänomen durch Stratifizierung der Daten entgegenzuwirken — man wählt die Mengen so, dass die Verhältnisse zwischen den Klassen ähnlich sind, dies schließt aber nicht aus, dass andere, weniger sichtbare, Korrelationen zwischen den Mengen erhalten bleiben. Die empirischen Vergleichsstudien [46] und [16] liefern teilweise widersprüchliche Ergebnisse dazu, ob Kreuzvalidierung oder Bootstrap-Heuristik, die jetzt von uns nicht näher behandelt wird, besser abschneiden. In dieser Arbeit wird Kreuzvalidierung benutzt, da sie auf optimistischen Bias weniger anfällig ist [46]. Allgemein kann festgestellt werden: Bis es eine klare theoretische Basis gibt, die stichfeste Ergebnisse bezüglich der Heuristiken gibt, bleibt angesichts der teils widersprüchlichen Studien nichts anderes übrig, als die Heuristiken trotzdem anzuwenden und das Risiko einzugehen, dass die Ergebnisse sich, nachdem man ihr Verhalten erforscht hat, als nicht valide erweisen. Dies wird auch von anderen Forschern sehr häufig gemacht.

3.1.2 Gefahren der statistischen Auswertung

Damit eine solche Auswertung glaubwürdig ist, behandeln wir jetzt die möglichen Fehler, die dabei häufig gemacht werden [71]. Ein häufiges Merkmal von Lernalgorithmen ist es, dass sie viele Parameter haben. Wenn man nun lediglich eine Testmenge und eine Trainingsmenge hat, mit denen man die verschiedenen Parameter anpasst, um ein gewisses Performanzmaß eines Verfahrens zu verbessern, läuft man Gefahr, dass der ermittelte Wert viel zu optimistisch ist. Dies ist eine verhüllte Art des Trainierens mit der Testmenge. Solche Probleme können vermieden werden, wenn man mit unterschiedlichen Daten trainiert und testet. Sie sind auch weniger akut, wenn das benutzte Verfahren wenig Parameter hat, die sich stabil auf das Ergebnis auswirken³. In dieser Kategorie sind zum Beispiel lineare SVMs, wo nur der Wert von C angepasst wird. In dieser Arbeit erfolgte zwar die Parameteranpassung mit den gleichen Daten wie die Auswertung, doch es waren immer nur eine bis drei numerische Größen, die aus im voraus gesetzten Mengen von einigen wenigen Möglichkeiten gewählt wurden. Die Ermittelten Werte von Parametern für verschiedene Experimente waren sich oft sehr ähnlich. Ein zweites Problem betrifft die Situation, wo man eine Aufstellung von mehreren Klassifikationsverfahren konstruiert und erfahren möchte, ob ein gewisses Verfahren besser abschneidet als ein gegebener Referenzwert. Da die Anzahl der getesteten Verfahren typischerweise groß ist⁴, wird man zu falschen Schlüssen verleitet, wenn man jedes der erzielten Ergebnisse mit einem Referenzwert vergleicht, der aus einem Konfidenzintervall für ein einmaliges Experiment resultiert (Multiplizitätseffekt). Ähnliche Problem können auftreten, wenn eine größere Anzahl von Forschern das gleiche stochastische Experiment durchführt oder eine gemeinsame Datenbank benutzt (Data-Snooping). Eine Möglichkeit, Multiplizitätseffekten entgegenzuwirken, ist die Bonferroni-Korrektur, wo man davon ausgeht, dass die Experimente unabhängig sind⁵ und die Schranke enger gemacht wird. Die Bonferroni-Korrektur ist so konservativ, dass sie meistens nicht praktikabel ist. Es ist theoretisch möglich, dass für den korrelierten Fall bessere (engere) Schranken hergeleitet werden, es sind dem Autor aber dazu keine Ergebnisse bekannt. In dieser Arbeit wurde das Problem der Multiplizität dadurch gemindert, dass auch die schlechten Ergebnisse immer angegeben

³Das heißt, dass eine kleine Änderung des Parameters eine kleine Änderung des Ergebnisses verursacht.

⁴Sie bestehen ja aus mehreren Phasen (z. B. Extraktion, Merkmalsuche usw.), wobei man im Prinzip jede Kombination betrachten kann.

⁵Im Bereich des Maschinellen Lernens sind sie es aber offensichtlich nicht.

wurden.

3.1.3 Der ROC-Raum

Problematisch bei der Anwendung der Trefferquote als Performanzmaß ist die Tatsache, dass sie die Kostenverteilung der Zielfunktion nicht berücksichtigt. Es kann sein, dass für den Anwender des Klassifizierers die Kosten der falschen Klassifizierung unterschiedlich sind, je nachdem was der Wert dieser Klassifizierung ist und dass sie dem Forscher überhaupt nicht zugänglich sind. Dies wird umso wichtiger, wenn der benutzte Datensatz bereits so vorbereitet worden ist, dass sich die Klassen in einem bestimmten Verhältnis zueinander befinden⁶. Deswegen ist es wichtig, ein Performanzmaß zu finden, das die ganze Breite von Kostenfunktionen berücksichtigt. Die in der Literatur oft geäußerte Behauptungen, dass ein Klassifizierer, der maximale Trefferquote aufweist, in praktischen Fällen ebenfalls die Kostenfunktion minimiert oder, dass er so umgebaut werden kann, dass dies erreicht wird kritisieren die Autoren von [64] als nicht genug begründet. Eine Lösung der angezeigten Probleme ist es, binäre Klassifizierer in dem ROC-Raum⁷ zu vergleichen, das heißt das Verhältnis der Fälle, wo ein negativ etikettiertes Objekt als positiv erkannt wird (FP) zu der Anzahl der Objekte, die zurecht als positiv identifiziert wurden (TP). Ein Durchlauf des Algorithmus entspricht einem Punkt in diesem Raum; wenn der Algorithmus Vertrauenswerte ausgibt, kann man durch das entsprechende Setzen der Schranke eine ganze Kurve zeichnen. Der entscheidende Vorteil dieser Kurve im Vergleich zur globalen Trefferquote liegt darin, dass sie von der Kostenfunktion und der Verteilung der Zielvariable unabhängig ist⁸. Wenn man mehrere Klassifizierer miteinander vergleicht, so kann es sein, dass eine der Kurven dominiert, das heißt alle andere Kurven befinden sich entweder unter ihr oder überschneiden sich mit ihr. In einem solchen Fall kann man sagen, dass der Klassifizierer, der der Kurve entspricht, unabhängig von dem Kontext der beste ist. In einem solchen Fall wäre auch die Messung der Trefferquote ausreichend⁹ Untersuchungen von Provost et. al. [64] ergaben aber, dass solche Fälle für praktische Probleme selten sind und dass es oft zwei oder mehr Klassifizierer gibt, die in verschiedenen Gebieten des ROC-Raumes lokal dominieren. Provost et. al. [64] ziehen auch eine andere Studie [15] heran, die ebenfalls zu diesem Schluss führt (bei leicht veränderter Methodologie). Dieser Umstand wird noch dadurch zusätzlich verschärft, dass die Kurven in der Praxis auch nur mit begrenzter Genauigkeit ermittelt werden können. Dann muss der Unterschied statistisch signifikant sein.

Es stellt sich natürlich sofort die Frage, wie ROC-Kurven zu generieren sind. Nur ein Experiment durchzuführen ist nicht ausreichend — das entspricht dem Holdout-Ansatz, dessen Schwächen wir bereits besprochen haben. Der häufige Ansatz ist es, Kreuzvalidierung durchzuführen, wobei es mehrere Möglichkeiten geben kann, die generierten Kurven zu mitteln. Dabei ist auch nicht der berechnete Durchschnitt wichtig, sondern auch die Varianz. In [32] wurden die zwei einfachsten Ansätze beschrieben. So kann man sie ROC-Kurve als eine Funktion betrachten (bei Konflikten wird der größte Wert genommen) und die TP-Werte mitteln oder aber längs der Kurven den Schwellenparameter variieren und für einen gegebenen Wert davon die Menge der Punkte betrachten, die auf den zu mittelnden Kurven liegen und diesem Wert entsprechen und den zweidimensionalen Mittel dieser Menge zu ziehen. Im

⁶In einem solchen Fall ist die reine Trefferquote komplett wertlos — wir lösen einfach ein anders Problem [64].

⁷Sie Abkürzung ROC kommt von receiver operating characterisitc — die Methode wurde zuerst in der Signalverarbeitung genutzt.

⁸Dies ist deshalb so, weil die Beiden betrachteten Variablen nur von dem Verhalten des Klassifizierers für Objekte einer konkreten Klasse abhängen.

⁹Man kann es aber nur dann sagen, wenn die Kurve schon gezeichnet ist, nicht *a priori* nur anhand der Trefferquote. Deswegen beschäftigt man sich ja mit den ROC-Kurven!

ersteren Fall enthält man eindimensionale Vertrauensintervalle auf der TP-Achse, im letzteren sind die Intervalle Rechtecke. Es gibt auch spezielle Methoden zur direkten Berechnung von ROC-Bändern. Da sie für die jetzige Arbeit nicht benutzt werden, werden sie hier auch nicht näher besprochen. Bei dem Mitteln von ROC-Kurven mit Schwellenwerten ist noch eines zu bemerken: Man darf auf keinen Fall Vertrauenswerte miteinander vergleichen, die von verschiedenen Algorithmen generiert wurden, wenn diese andere Skalen benutzen. Es bleibt noch die Frage offen, wie die Paare (TP, FP) für SVMs generiert werden können, also für einen Klassifizierer, der nur Binäre Antworten liefert. Dies wurde erreicht, indem die SVM nach dem Vorschlag von [62] um die Einschätzung von Wahrscheinlichkeiten der jeweiligen Klasse erweitert wurde. Zur Sigmoid-Anpassung wurde die MATLAB-Funktion `glmfit` verwendet. Wenn die Wahrscheinlichkeitseinschätzung bekannt ist, können mittels der Verschiebung des Schwellenwerts verschiedene Punkte der Kurve generiert werden.

Ein weiteres interessantes Merkmal des ROC-Ansatzes ist es, dass sie, in dem Fall, wenn wir die Kostenfunktion des betrachteten Problems kennen, zwei Punkte des ROC-Raumes miteinander vergleichen können [63]. Nehmen wir nur an, dass die Konstante c_{FP} der „Preis“ dessen ist, dass der Klassifizierer ein negatives Objekt als positiv identifiziert hat; c_{FN} der fälschlicher Identifizierung als negativ entspricht und die Wahrscheinlichkeiten $P(P)$ und $P(N)$ der Verteilung der Zielvariable. Der zu erwartende Kostenpunkt für das Element (FP_1, TP_1) des ROC-Raumes ist wie folgt.

$$P(P)(1 - TP_1)c_{FN} + P(N)(FP_1)c_{FP}$$

Wenn wir nun verlangen, dass der Kostenpunkt für zwei Paare (FP_1, TP_1) und (FP_2, TP_2) gleich ist, bekommen wir die Gleichung.

$$\frac{TP_2 - TP_1}{FP_2 - FP_1} = \frac{P(N)c_{FP}}{P(P)c_{FN}}$$

Dies bedeutet, dass bezüglich der gemachten Annahmen der Bereich des ROC-Raumes, wo der Klassifizierer dominiert, der den Punkt (FP_1, TP_1) generiert hat, sich unter der Linie befindet, die durch diesen Punkt verläuft und die Steigung $(P(N)c_{FP})/(P(P)c_{FN})$ hat.

Eine Schwäche, die ROC-Analyse hat ist es, dass sie in der dargestellten Form nur für binäre Probleme geeignet ist. Bereits bei drei Klassen müsste die Kurve im sechsdimensionalen Raum definiert sein; für Regression sind dem Autor keine Verallgemeinerungen bekannt (es sei denn die Ausgaben lassen sich als Vertrauenswerte definieren).

3.1.4 Der teilüberwachte Fall

Es ist schwierig, für unüberwachtes und teilüberwachtes Lernen Techniken zur Beurteilung von Algorithmen zu definieren — bei Thurn et. al. [80] wird dies mit der Feststellung bezeichnet, dass es für teilüberwachtes Lernen keine mittlere Performanz gibt. So ist es nicht klar, was die Rolle der unetikettierten Daten ist — ob sie bei der Kreuzvalidierung wie die etikettierten aufgeteilt werden sollen, oder ob man stets mit der vollen Menge der Daten trainiert, und wenn ja, wie. Bei Transduktion ist das auch nicht einfacher. Kääriäinen [44] schlägt ein Verfahren vor, das auf der Idee basiert, die unetikettierten Daten in das Auswahlverfahren des Klassifizierers selbst zu integrieren: Sie werden zum Vergleich zwischen einem Einzelklassifizierer und einem Ensemble-Klassifizierer benutzt, der mehrere Einzelklassifizierer eingebaut hat und der Kreuzvalidierung entspricht, wobei es möglich ist, aus diesem Vergleich Schlüsse über die Performanz des Ensemble-Klassifizierers zu ziehen (in der Arbeit von Thurn et. al. [80] werden auch ähnliche Ideen behandelt). Schuurmans et. al. [73] untersuchen, inwiefern die unetikettierten Daten zur Berechnung der Abstände zwischen den

Hypothesen geeignet sind, wobei dies benutzt wird, um am Ende die geeignetsten Hypothesen auszuwählen (siehe auch Abschnitt 2.4.4). Diese Ansätze wurden in der jetzigen Arbeit mangels Zeit nicht untersucht, es wird stattdessen auf einfache Heuristiken zurückgegriffen.

3.1.5 Auswertungsgrößen bei Regression

Bei Klassifizierung ist die Suche nach einer einfachen und leicht zu interpretierenden Heuristik für die Auswertung der Pipeline schnell erledigt: Man nimmt einfach die Trefferquote. Bei Regression dagegen ist das weniger offensichtlich. Das Problem mit Standardmaßen wie MSE oder MAE ist, dass sie direkt von der Größenordnung der Etiketten abhängen, was einen Vergleich zwischen verschiedenen Datensätzen erschwert. Eine Lösung bietet der Q_2 -Score, der wie folgt definiert ist [69].

$$Q_2 = 1 - \frac{MSE}{MSE_{mean}}, \quad MSE_{mean} = n^{-1} \left(\sum_i^n (t_i^2 - (n^{-1} \sum_j^n t_j)^2) \right)$$

Der Q_2 kann nun so interpretiert werden: negative Werte, bedeuten, dass das Verfahren schlechter ist, als der Durchschnitt-Klassifizierer¹⁰, also sehr schlecht. Werte nah an 1 bedeuten, dass der MSE-Fehler sehr klein ist, das Verfahren also gut ist. Zum Zweck des Vergleichs mit Literaturergebnissen wurde in dieser Arbeit auch eine Normierte Version von MSE, der RMS-Score [95] verwendet: $RMS = \sqrt{n^{-1}MSE}$.

3.2 Das benutze Auswertungsverfahren

Wie sich aus den vorgängigen Abschnitten ergibt, sind alle praktikablen Methoden der Auswertung von Klassifizierern Heuristiken, deren Verhalten im überwachten Fall nicht ausreichend und im teilüberwachten nahezu gar nicht erforscht ist — es gibt bisher keine Theorie der Validierung, die auf praktische Probleme anwendbar wäre. Deswegen muss auch in dieser Arbeit auf Heuristiken zurückgegriffen werden. Es wurde also die folgende Methodologie angewandt.

1. Die Verfahren werden wenn möglich mit zehnfacher Zehn-Fold-Kreuzvalidierung getestet; wenn dies nicht möglich ist, da man mehr unetikettierte Daten braucht, wird randomisiertes 20-faches Holdout benutzt. Die Trefferquote dient als „rohes“ Performanzmaß zur Einschätzung der Situation. Auf diese Weise entstehen Lernkurven.
2. Für zwei ausgewählte Punkte der Lernkurve werden, wo ein Vergleich im ROC-Raum sinnvoll ist, zusätzlich ROC-Kurven generiert.
3. Die Parameter für Lernverfahren werden wie folgt gewählt: Bei SVM ist C immer 1, bei kNN wird k auf 20 gesetzt. Diese Parameter werden nicht getunt. Bei Auswertung der Merkmalsuche (Lernkurven 5 und 6) werden die Werte der Parameter aus den folgenden Mengen ausgewählt: die Anzahl der Merkmale aus 10, 50, der Balanzierkoeffizient β aus 0, 0.5, 1, 2 und die Sprungwahrscheinlichkeit bei stochastischer Suche aus 0.1, 0.2, 0.5.
4. Aus Zeitgründen (da TSVMs bei vielen unetikettierten Daten langsam sind), werden die Lernkurven 16, 17 und 19 nur mit einer Untermenge des jeweiligen Datensatzes berechnet (Details siehe Kurve). In der Lernkurve 5 wurden die Wrapper-Ergebnisse mit einer Untermenge berechnet.
5. Bei Literaturvergleichen wird immer die Methodologie des Quellpapers verwendet.

¹⁰Es gibt in der Literatur verschiedene Ansätze dazu, ob der Mittelwert jeweils nur mit der Trainingsmenge oder mit allen Etiketten berechnet werden soll. In dieser Arbeit wird er nur anhand der Trainingsmenge berechnet.

Tabelle 3.1 Parameter für Regressionsverfahren (DR - direkte Regularisierung, RF - Regularisierung in einer Funktionsfamilie).

Datensatz	SVM	DR ^c	RF
EDKB	$C = 10, nu = 0.1$	$C_L = 100,$ $C_{mult} = 10^{-8}, \sigma = 0.02$	$\gamma_A = 10^{-2}, \gamma_I = 0.1,$ $\sigma = 0.5$
BERGSTROM	$C = 1, nu = 0.2$	$C_L = 10^3,$ $C_{mult} = 10^{-6}, \sigma = 0.04^b$	$\gamma_A = 10^{-2}, \gamma_I = 0.1,$ $\sigma = 0.5$
PDGFR	$C = 1, nu = 0.5$	$C_L = 100,$ $C_{mult} = 10^{-6}, \sigma = 0.05$	$\gamma_A = 0.01, \gamma_I = 10^{-3},$ $\sigma = 0.05$
EPA	$C = 1, nu = 0.2$	$C_L = 10^4,$ $C_{mult} = 10^{-8}, \sigma = 0.05$	$\gamma_A = 0.01, \gamma_I = 1,$ $\sigma = 0.1^b$
EPA norm. (A) ^a	$C = 1, nu = 0.5$	–	–
EPA norm. (B) ^a	$C = 10, nu = 0.2$	–	–
COX	$C = 1, nu = 0.2$	–	–
COX norm. (A) ^a	$C = 10, nu = 0.5$	–	–
COX norm. (B) ^a	$C = 10, nu = 0.5$	–	–

^a Die Abkürzung norm. bedeutet die normierte Version des Datensatzes (eine mit möglichst flachem Histogramm), wobei zur Auswertung entweder die ursprünglichen (A) oder die transformierten (B) Etiketten verwendet wurden.

^b Für die BERGSTROM und EPA-Datensätze wurde versucht, die Ergebnisse manuell anzupassen. Da die Q_2 -Werte aber ohnehin schlecht sind, verletzt das nicht die angenommene Methodologie.

^c $C_{UL} = C_L C_{mult}$

Bei allen Schwächen der Trefferquote hat sie den Vorteil, dass sie einfach zu berechnen ist. Die Randomisierung bietet eine bequeme Möglichkeit, die unetikettierten Daten miteinzu-beziehen: Es wird jeweils eine Permutation der Moleküle generiert, die die Objektmenge in vier Gruppen, deren Größe von dem Benutzer spezifiziert wird, unterteilt: die etikettierten und unetikettierten Graphen, sowie die, die zum Testen dienen und die, die weggeworfen werden¹¹.

Bei Regression wird zum Testen 5-malige 5-Fach Kreuzvalidierung benutzt. Sie wird verwendet, da die EDKB- und PDGFR-Datensätze wesentlich kleiner sind als die bei Klas-sifizierung und die Testmenge bei 10 Folds zu klein wäre. Als gemessene Größe wurde der Q_2 -Wert verwendet, da er einfacher zu interpretieren ist als MSE und auch einen Vergleich zwischen unterschiedlichen Datensätzen ermöglicht. Andere Auswertungsverfahren werden zu Literaturvergleichen verwendet, sie sind in der Tabelle 3.7 jeweils direkt angegeben. Die Parameter für Regressionsverfahren werden wie folgt gewählt. Für SVMs wird ein C aus der Menge 0.1, 1, 10 und ein Nu aus der Menge 0.1, 0.2, 0.5, 1.0 genommen. Für kNN wird k auf 5 oder 50 gesetzt. Für direkte Regularisierung wird σ aus der Menge 0.05, 0.2, 0.5, 2 genom-men, C_{mult} aus der Menge $10^{-1}, 10^{-2}, \dots, 10^{-8}$ und C_L aus $10^0, \dots, 10^6$ ($C_{UL} = C_L C_{mult}$). Für Regularisierung mit Kernels wird σ gleich wie früher ausgewählt, γ_A kommt aus der Menge 1, 0.1, 0.01 und γ_I aus der Menge $10^0, \dots, 10^{-6}$. Die optimalen Werte der Parameter sind in der Tabelle 3.1 zu sehen.

¹¹Dies ist nötig, um Varianz zu kontrollieren, da die Klassifizierer nicht stabil sind.

Tabelle 3.2 Datensätze zur Klassifizierung

	Moleküle	Klasse		Support ^b
		aktiv	inaktiv	
NCTRER	224	58%	42%	10%
BBP2	408	66%	34%	10%
FONTAINE	435	64%	36%	10% / 20% ^a
CYP 2D6-inhib	} 695	74%	26%	} 10%
CYP 2D6-sub		72%	28%	
CYP 2A4-inhib		66%	34%	
CYP 2A4-sub		48%	52%	
CYP 2C9-inhib		76%	24%	
CYP 2C6-sub		79%	21%	

^a Für Experimente mit der Verbindung von Supportschränke und χ^2 (Lernkurven 7,21 und 22) wurde 10% verwendet, ansonsten 20%.

^b Support von 10% bedeutet, dass mindestens 10% des Moleküle einen Untergraphen beinhalten müssen. Mehrmaliges Vorkommen des Untergraphen in einem Molekül wird ignoriert. Es werden nur geschlossene Untergraphen verwendet.

3.3 Die benutzten Datensätze

Es wurden zur Validierung der Merkmalsuch- und Klassifizierungsansätze die in der Tabelle 3.2 genannten Datensätze benutzt. Es wird auch jeweils in Großbuchstaben eine Abkürzung angegeben, die später in den Grafiken benutzt wird.

1. Die NCTRER-Datenbank [86] (National Center for Toxicological Research - estrogen receptor). Sie beinhaltet Informationen, wie gut sich bestimmte Stoffe an Estrogenrezeptoren binden. Es wurden aus der SDF-Datei nur das Feld `ActivityOutcome_NCTRER` und die Molekülgraphen benutzt. Moleküle, wo der Wert „inconclusive“ war, wurden weggelassen.
2. Der BBP2-Datensatz [53] beinhaltet Informationen darüber, ob bestimmte Stoffe die Blut-Hirn-Schranke durchdringen können (aktiv bedeutet ein erfolgreiches Durchdringen).
3. Der FONTAINE-Datensatz [34] beinhaltet Faktor Xa-Inhibitoren. Die in der Originalpublikation angegebenen Test- und Trainingsmengen wurden in einen Satz zusammengefasst. Da in der Datensatz nur sehr aktive Stoffe enthält (keine Zwischenstufen), ist er eine relativ leichte Aufgabe für Klassifizierung.

Zur Beurteilung der Regressionsalgorithmen wurden folgende Datensätze benutzt (siehe Tabelle 3.3).

1. Die EDKB-Datenbank¹². Es wurden nur die Ergebnisse des E-Screen Verfahrens ausgewertet. Die Untermenge EDKB/S wird verwendet, um Ergebnisse mit Saigo et. al. [70], [69] vergleichen zu können. Es wird nur das Feld `LogRBA` verwendet.
2. Der BERGSTROM-Datensatz [8] beinhaltet Angaben zur Schmelztemperatur verschiedener Stoffe.
3. Der EPA-Datensatz [66]. Es wurde die Toxizität für *pimephales promelas* verwendet, die dem Feld `LC50_mg` der Datenbank entspricht.

¹²Siehe <http://edkb.fda.gov>

Tabelle 3.3 Datensätze zur Regression

	Moleküle	Untergraphen	Support ^a
EDKB	80	1881	5%
EDKB/S ^b	59	2452	alle
BE	277	2296	5%
EPA	580	729	2%
COX	414	976	20%
PDGFR	79	318	alle
PHENOL	153	450	alle

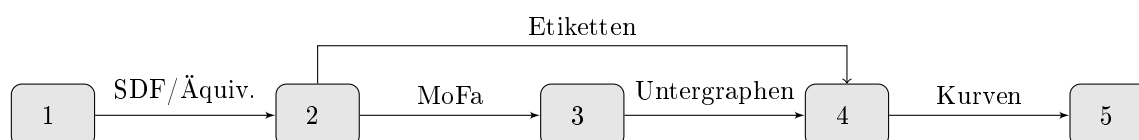
^a Die Zahl bedeutet den Anteil an Molekülen, die den Untergraphen beinhalten müssen, „alle“ bedeutet, dass alle geschlossenen Untergraphen berücksichtigt werden.

^b EDKB/S ist eine verkleinerte Version des EDKB-Datensatzes. Da dieser mit der Zeit erweitert wurde, muss für Vergleiche mit älteren Papers die entsprechende Untermenge verwendet werden.

4. Der COX-Datensatz [79] beinhaltet Informationen zu Inhibitoren von Cyclooxygenase-2. Der Aktivitätswert entspricht dem Feld IC50_uM im Datensatz.
5. Der PHENOL-Satz [95] beinhaltet Angaben zur Toxizität diverser Phenole für den tetrahymena-pyryformis-Protist.

3.4 Die Pipeline

Abbildung 9 Die einzelnen Schritte der Pipeline.



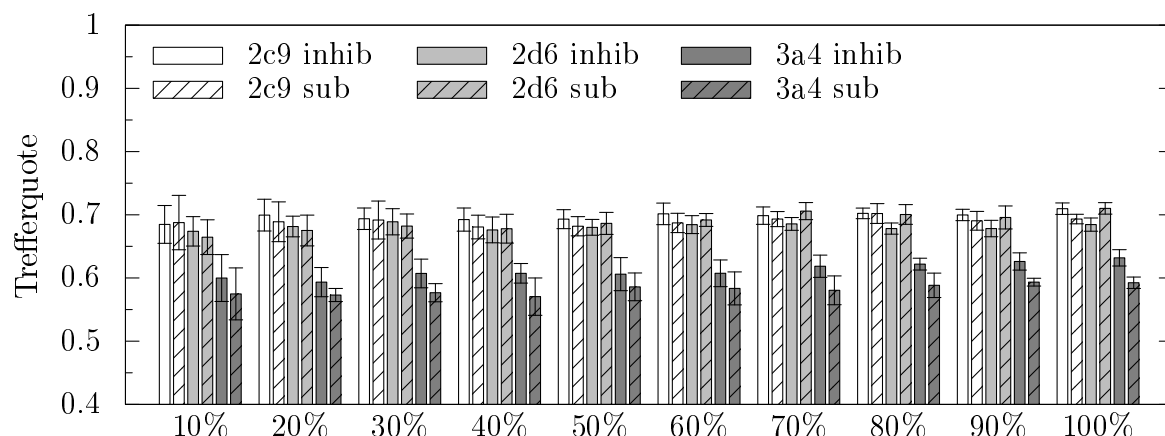
1. Textuelles Preprocessing.
 2. Konversion, es entsteht eine Graphendatei im MoFa-Format und eine mit der Etikettierung.
 3. Untergraphenextraktion (nur geschlossene Graphen) mit MoFa.
 4. Merkmalsuche und Klassifizierung oder Regression.
 5. Die Erzeugten Kurven werden mit `gnuplot` visuell Verarbeitet.
-

In diesem Abschnitt werden die Schritte erläutert, die von den Eingabedateien zum Endergebnis führen. Sie sind in der Abbildung 9 zu sehen. Da die Datensätze in unterschiedlichen Formaten vorlagen, mussten sie vor der Anwendung entsprechend konvertiert werden. SDF-Dateien können direkt eingelesen werden, andere Formate werden mit jeweils einmalig angepassten Skripten manuell konvertiert. Dann wird die SDF-Datei in eine Java-Anwendung eingespeist, die auf `joelib` basiert, und eine Eingabedatei für MoFa sowie eine Datei mit

Tabelle 3.4 Implementierungen verschiedener Klassifizierungs- und Regressionsverfahren.

Klassifizierer	Implementierung	
SVM (linearer Kernel)	<code>liblinear</code>	} Klassifizierung
SVM (RBF-Kernel)	<code>weka.classifiers.functions.SMO</code>	
TSVM / SVM ^a	Matlab-Code von Chapelle et al. [20]	
kNN	Eigenimplementierung	
j48	<code>weka.classifiers.trees.j48</code>	
reg-SVM	<code>libsvm.svm</code>	} Regression
kNN-Reg	Eigenimplementierung	
direkte Regularisierung	Eigenimplementierung	
Regularisierung mit Kernels	Eigenimplementierung	

^a Bei Vergleichen SVM vs. TSVM wurde die gleiche Implementierung benutzt, um Unterschiede in den Details der Implementierung auszuschalten.

Ergebniskurve 1 Lernkurven für CYP-Datensätze (SVM). 10-fache 10-Fold CV, Prozente bedeuten Anteil des Folds.

Etiketten erzeugt. Dabei werden alle Atome und chemischen Bindungen mit numerischen Werten ersetzt. Anschließend erfolgt die eigentliche Extraktion der Untergraphen, wobei die Parameter `-inelist -flinog -onelist -s<Support>` benutzt werden. Der MoFa-Miner gibt zwei Dateien aus, die eine enthält Informationen, in welchen Graphen welcher Untergraph vorkommt, die zweite enthält die Definitionen der Untergraphen. Fürs Lernen braucht man nur die erstere, die zweite wird nicht weiterverwendet. Dann werden die Zugehörigkeitsinformation der Untergraphen und die Etikettierung in eine andere Java-Anwendung eingespeist. Dort erfolgt Merkmalsuche, Klassifizierung oder Regression und die Generierung der Ergebniskurven. Die Tabelle 3.4 zeigt, was selber implementiert wurde und an welcher Stelle fertige Bibliotheken benutzt wurden. Die Ausgaben wurden dann mit der `gnuplot`-Anwendung geplottet.

3.5 Interpretation der Ergebnisse - Klassifizierung

Die Lernkurven 1,12,13 deuten darauf hin, dass sich der CYP-Datensatz unabhängig von dem benutzten Klassifizierer und der gewählten Etikettierung nur schwer mit den hier beschriebenen Werkzeugen erfassen lässt. Die Generalisierungsfähigkeit ist sehr schlecht — man könnte sogar sagen, dass sie sich außer bei kNN, wo sie schwach ist (siehe Lernkurve

Tabelle 3.5 Vergleich der Klassifizierungsergebnisse mit Yap und Chen (Tabelle 4 in [96]).

Klass.	Parameter ^a	diese Arbeit ^f			Yap und Chen ^d			Auswertung
		<i>SN</i>	<i>SP</i>	<i>ACC</i>	<i>SN</i>	<i>SP</i>	<i>ACC</i>	
SVM ^b	$C = 1$	0.64	0.84	0.75	0.98	0.85	0.92	} einmalig ^c
kNN ^h	$k = 20$	0.75	0.93	0.85	0.92	0.83	0.88	
j48 ^e	—	0.70	0.88	0.80	0.76	0.66	0.72	

^a Die Parameter beziehen sich auf diese Arbeit, die Parameter, die von Yap und Chen benutzt wurden, sind unbekannt. Merkmalsuche wurde nicht benutzt. Es wurden alle Untergraphen mit 10% Support oder mehr berücksichtigt.

^b Es wurde hier linearer Kernel benutzt. Radialer Kernel brachte keine Verbesserung. Der von Yap und Chen verwendete Kernel ist unbekannt.

^c Es wurden die von Yap und Chen vorbereiteten Test- und Trainingssätze verwendet (CYP 2A4 - Substraten). Die Sätze sind nicht unabhängig, sondern wurden gezielt so konstruiert, dass der Trainingssatz aussagekräftig im Bezug auf den Testsatz ist.

^d Yap und Chen geben Ergebnisse an, die sich aus 30 Durchläufen eines stochastischen Merkmalsuchverfahrens ergeben. Es wird hier nur der Mittelwert angegeben.

^e Von Yap und Chen wurde C4.5 (ohne Grafting) benutzt.

^f Abkürzungen: $SN = TP/(TP + FN)$, $SP = TN/(FP + TN)$, ACC ist Trefferquote.

^h Die Nachbarschaftsfunktion, die von Yap und Chen verwendet wurde, ist unbekannt. In dieser Arbeit wurde der Tanimoto-Abstand verwendet.

12), gar nicht beobachten lässt, zumal die große Varianz der Ergebnisse in Anfangsteilen der Kurven in Betracht gezogen wird. Warum das so ist, kann man nicht mit Sicherheit sagen. Eine der Möglichkeiten ist die Auswahl der negativen Beispiele: Da es sehr schwierig zu beweisen ist, dass ein Stoff nicht aktiv ist, wird es von Yap und Chen [96] dann angenommen, wenn der Stoff in den von ihnen untersuchten Studien nicht als Inhibitor, Substrat oder Agonist eines anderen Proteins identifiziert wurde. Dies ist problematisch, da sich der Stoff doch als aktiv erweisen kann. Es kann auch sein, dass der induktive Bias der Klassifizierung falsch ist (der zum Beispiel bei SVMs, wo man die von der theoretischen Seite her höchstens fragwürdige Transformation von dem Graphenraum nach \mathbb{R}^n macht, gar nicht in einfachen, mit dem Aufbau der Molekülen unmittelbar korrespondierenden Begriffen erfassen lässt). Möglich ist auch, dass die Datensätze zu klein sind: Wenn sich in der Testmenge Moleküle befinden, wo durch gewisse Untergraphen eine Eigenschaft vorhanden ist, die aber nicht in der Trainingsmenge zu beobachten sind, kann es natürlich keinen Klassifizierer geben, der so ein Molekül außer als durch Zufall richtig klassifiziert. Auch wenn wir davon ausgehen würden, dass alle „abgefragten“ Eigenschaften in der Trainingsmenge vorhanden sind, kann es dennoch zu Problemen kommen, wenn die Zielvariable durch beliebige Boolesche Zusammensetzung der Untergraphen-Indikatorvariablen zustande kommt: In einem solchen Fall würde man natürlich exponentiell viel Trainingsobjekte brauchen. Es kann schließlich auch sein, dass der gewählte Support zu groß war.

Um sicherzustellen, dass die schlechten Ergebnisse für den CYP-Datensatz tatsächlich durch die Beschaffenheit der gewählten Ansätze verursacht sind und nicht etwa durch einen Pipeline-Fehler, wurden sie mit denen von Yap und Chen [96] verglichen (siehe Tabelle 3.5). Es wurden die von Yap und Chen konstruierten Training- und Testsätze verwendet (obwohl sie nicht unabhängig waren), deswegen konnte das Experiment nur einmal wiederholt werden. Es wurde nicht versucht, das genetische Merkmalsuchverfahren von Yap und Chen nachzubauen, die erreichten Ergebnisse kamen ohne Merkmalsuche zustande. Dabei ist auch anzumerken, dass die SVM-Ergebnisse nicht ganz vergleichbar sind, weil hier von Yap und Chen wahrscheinlich (im Paper gibt es dazu keine Angaben) eine Kombination von mehre-

Tabelle 3.6 Vergleich der Klassifizierungsergebnisse mit Buchwald et al. (Tabelle 3 in [17]).

Klass.	Parameter	ACC^a	ACC	Datensatz
		gemessen	in [17]	
SVM	$C = 1, RBF$	0.96 (0.03)	0.84 (0.04)	} FONTAINE ^b
kNN	$k = 40$	0.88 (0.05)	0.85 (0.03)	
kNN	$k = 3$	0.95 (0.03)	0.94 (0.02)	
j48	—	0.93 (0.04)	0.88 (0.03)	
SVM	$C = 1, RBF$	0.77 (0.04)	0.79 (0.02)	} CYP 2C9-inhib ^b
kNN	$k = 10$	0.76 (0.05)	0.78 (0.02)	
kNN	$k = 3$	0.74 (0.05)	0.74 (0.03)	
j48	—	0.73 (0.05)	0.73 (0.03)	

^a ACC ist Trefferquote. Zahlen in Klammern sind Standardabweichungen. Als Auswertungsverfahren wurde Leave-one-out-Holdout mit 100-facher Wiederholung benutzt.

^b Die benutzte Supportschranke war für beide Datensätze 10%.

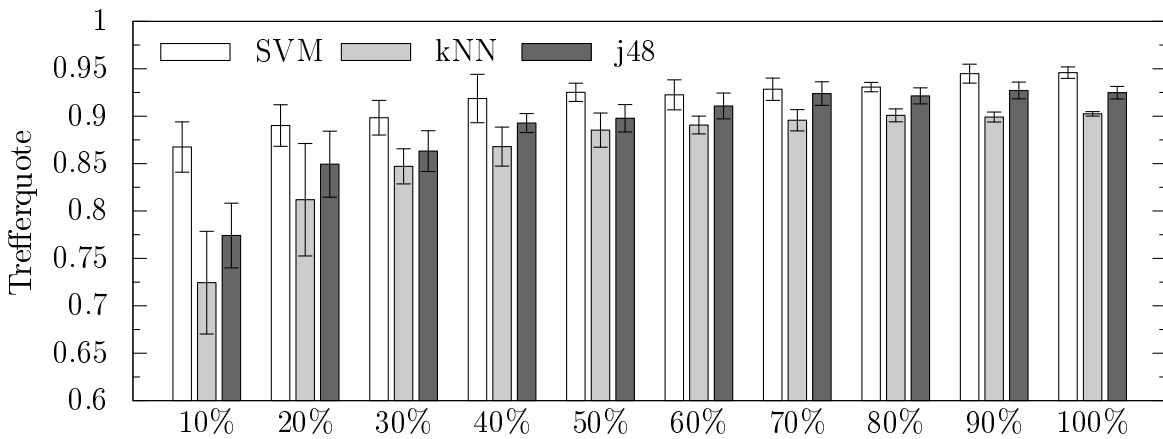
ren SVMs mit einfachem Boosting (CSVM) benutzt haben und keine herkömmliche SVM. Die restlichen Ergebnisse sind dagegen durchaus vergleichbar, wobei die SN- und SP-Werte darauf hindeuten, dass sich beide Experimente in einem anderen Punkt des ROC-Raumes befinden: Bei den gemessenen Werten ist im Vergleich zu Yap und Chen immer eine kleinere Sensitivity und eine größere Specificity feststellbar.

Die erzielten Ergebnisse wurden auch mit denen von Buchwald et al. [17] verglichen (siehe Tabelle 3.6). Auch hier kann man sagen, dass die Pipeline besteht: Die errechneten Werte sind für den stabilen FONTAINE-Datensatz wesentlich besser, für den CYP-Datensatz, der ohnehin problematisch ist, sind sie vergleichbar. Der größte große Unterschied ist bei SVM anzutreffen, wo für den FONTAINE-Datensatz die gemessene Trefferquote um mehr als 10% besser ist. Dies ist wahrscheinlich deswegen der Fall, weil in dieser Arbeit nur geschlossene Untergraphen benutzt wurden — eine Menge, die die Vollständigkeit der Information garantiert und Redundanz zumindest teilweise verhindert. So oder so ist die benutzte Pipeline insgesamt glaubwürdig.

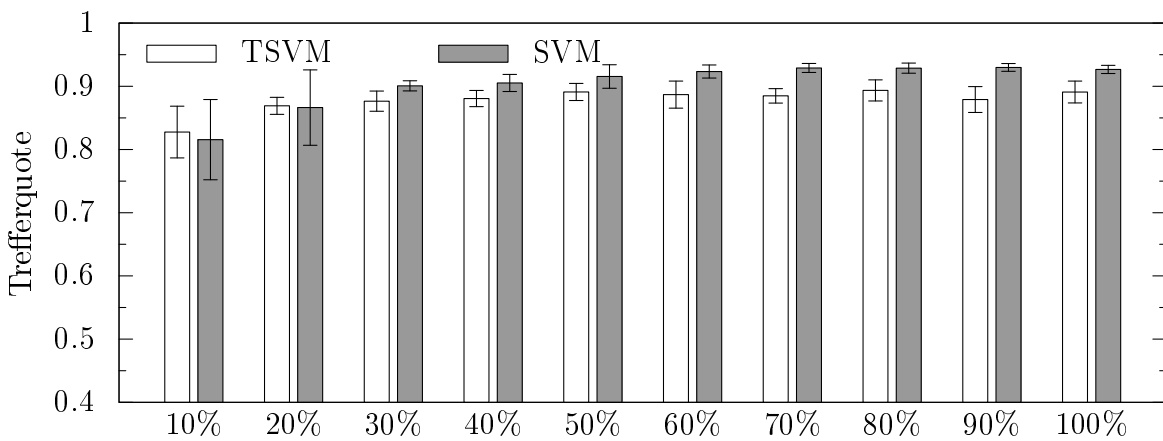
Die Ergebnisse mit dem BBP2-Datensatz waren recht mager (siehe Lernkurve 15): Die große Varianz in dem Anfangsteil der Kurve macht es schwierig, überhaupt Verallgemeinerungspotenzial festzustellen. Mit den FONTAINE- und NCTREER-Datensätzen waren die Ergebnisse dagegen besser (siehe Lernkurven 14 und 2), der Aufstieg der Kurve ist deutlich erkennbar. Da das absolute Ergebnis für den FONTAINE-Datensatz besser war, wurde dieser Datensatz für weitere Experimente verwendet.

Als nächstes wurde dieser Datensatz mit TSVMs getestet. Es ergeht klar aus Lernkurve 3, dass Transduktion keinen gewaltigen Vorteil nach sich zieht: Die Verbesserung der Trefferquote ist minimal. Dies geschieht, wie zu erwarten ist sowohl wenn man mit Kreuzvalidierung auswertet, als auch mit Holdout (siehe Lernkurve 3) — im letzteren Fall ist sogar ein kleiner Vorsprung vonseiten einer herkömmlichen SVM zu beobachten. Da TSVMs eine etablierte Methode sind, ist es in diesem Kontext interessant zu fragen, inwiefern sich diese fehlende Verbesserung auch bei anderen Kostenfunktionen als die symmetrische bemerkbar macht. Zu diesem Zweck wurden zwei ROC-Kurven (siehe Graphiken 4 und 18) erstellt, die zwei Punkten der Lernkurve (50% und 100%) entsprechen. Man sieht anhand den Kurven leicht, dass die beiden Ansätze gleichwertig sind: Alle Unterschiede sind kleiner als die ermittelten Werte von Standardabweichung der jeweiligen Messung. Eine andere Frage, die man im Kontext von TSVMs stellen kann ist, ob die zusätzlichen unetikettierten Graphen

Ergebniskurve 2 Lernkurven für den FONTAINE-Datensatz. 10-fache 10-Fold CV, Prozenzte bedeuten Anteil des Folds.



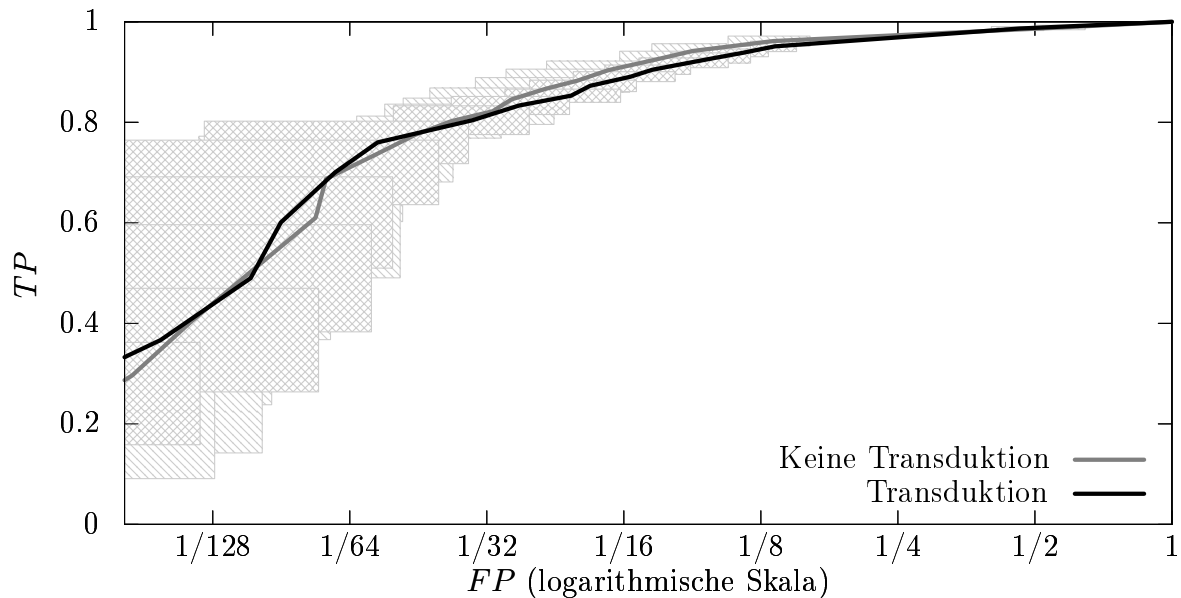
Ergebniskurve 3 Lernkurve für den FONTAINE-Datensatz, TSVM, Kreuzvalidierung. 10-fache 10-Fold CV, Prozenzte bedeuten Anteil des Folds.



vielleicht dessen ungeachtet eine bessere Entscheidungsregel ermöglichen. Aus Lernkurve 17 folgt, dass dies nicht der Fall ist: Wenn man dem Algorithmus unetikettierte Daten zur Verfügung stellt, die aber nicht den Testdaten entsprechen, ist keine Verbesserung im Vergleich zu einer üblichen SVM zu beobachten. Für andere (schwierigere) Datensätze sind die Ergebnissen unschlussig. Bei BBP2 ist keine Verbesserung nachweisbar, bei NCTRER sieht man eine leichte Verbesserung im Anfangsteil der Kurve (siehe Lernkurven 19 und 20).

In der Abbildung 5 werden Ergebnisse für Merkmalsuche mit Markovschen Decken, Inkonsistenzen und zwei Versuche mit Wrapper vorgestellt: Einmal der klassische Wrapper und einmal einer mit dem Balanzierkoeffizienten ungleich null (siehe Abschnitt 1.1.5), so dass auch unetikettierte Graphen miteinbezogen werden. Entgegen Erwartungen konnten sich die komplizierten Methoden gegen das klassische χ^2 -Maß nicht behaupten. Die Ergebnisse für Wrapper-Ansätze waren außer im Anfangsbereich der Kurve nur geringfügig besser, was insofern interessant ist als gerade diese Verfahren die Trefferquote eigentlich am meisten steigern sollen, da sie diese direkt optimieren; die unetikettierten Graphen bewirkten dabei keine Verbesserung der Trefferquote. Es ist teilweise dadurch verursacht worden, dass die Wrapper-Merkmalsuche nur mit einer Untermenge des Datensatzes getestet werden konnte. Andererseits erreichen hier die Kurven schnell Sättigung, es kann also sein, dass die Ergebnisse darauf zurückzuführen sind, dass die Score-Funktion für wenig relevante aber mit anderen unkorrelierte Merkmale und relevante aber gut korrelierte Merkmale ähnliche Werte annimmt. Die zwei anderen Verfahren, von denen Verbesserungen erwartet waren,

Ergebniskurve 4 ROC-Kurve: FONTAINE, TSVM vs. SVM (Training mit 50% des Folds, 10-fache 10-Fold CV). Die schattierten Bereiche entsprechen der Standardabweichung des (TP, FP) -Paares.

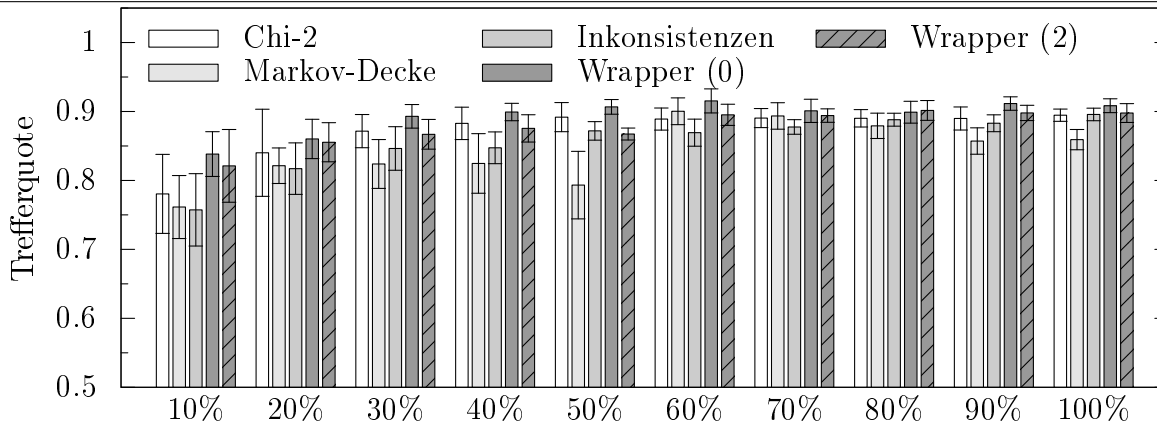


schnitten sogar schlechter ab, als χ^2 . Im Falle Markovscher Decken ist das wahrscheinlich auf die sehr imperfekte Art und Weise zurückzuführen, wie die Decke berechnet wird – der Begriff der Decke selbst hat eine gute theoretische Basis und ist stichfest. Bei Inkonsistenzen war wahrscheinlich die sehr einfache Definition der Score-Funktion schuld, die nur das Separiervermögen der Attribute berücksichtigt und nicht die eigentliche Relevanz.

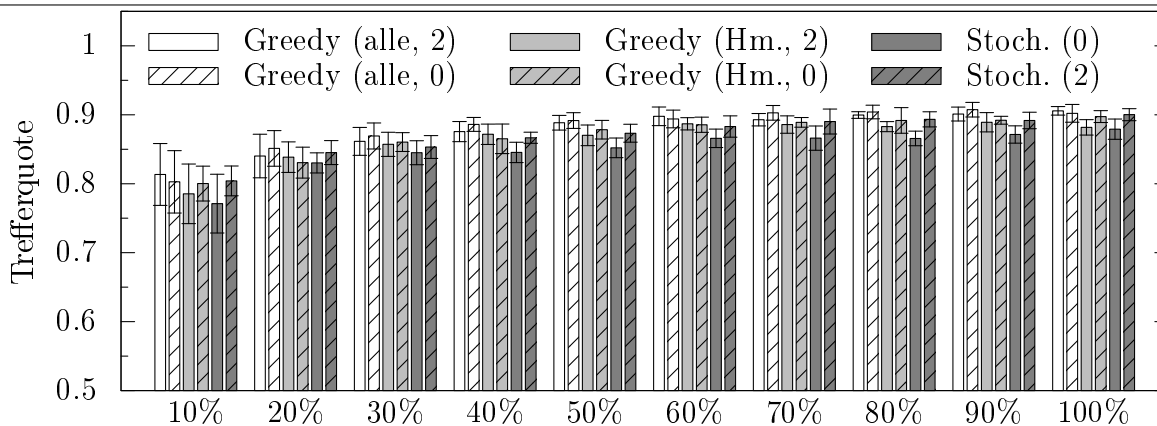
Anschließend wurden verschiedene Suchverfahren mit dem Skalarprodukt-Score ausprobiert (siehe Lernkurve 6). Dabei kann folgendes gesagt werden: Die komplizierte und zeitaufwändige stochastische Suche bringt keinen Vorteil, und schneidet zuweilen sogar schlechter ab, als die einfache Greedy-Suche. Bei Greedy-Suche gibt es dagegen wenig Unterschied zwischen der Situation, wo man in jeder Iteration alle möglichen Merkmale berücksichtigt oder nur 8 Merkmale mit dem größten Hamming-Abstand. Im Falle aller drei Suchverfahren ist dagegen zumeist ein kleiner Performanzgewinn zu beobachten, wenn die unetikettierten Daten miteinbezogen werden. Dieser ist jedoch zu geringfügig, als dass er als statistisch signifikant bezeichnet werden könnte.

Die Lernkurven 7, 21 und 22 stellen einen Versuch dar, einen Zusammenhang zwischen der Qualität der Support-Messung der Merkmale und der Trefferquote, die durch Klassifizierung mit Support-Filterung erreicht wurde, festzustellen. Die Idee ist in der Abbildung 10 zu sehen und besteht darin, relevante Merkmale nur unter denen zu suchen, die jeweils in der Menge der etikettierten Graphen, oder in der Menge aller Graphen, oft genug vorkommen. Dies entspricht den zwei schraffierten Bereichen der Abbildung 10. Technisch wurde dies als ein Zweischrittverfahren implementiert: Zuerst wurden die Merkmale mit zu kleinem Support eliminiert, dann wurden daraus die k ausgewählt, die am relevantesten waren (nach dem χ^2 -Maß). Experimente wurden mit Supportwerten von 10%, 20% und 30% und einem k von 5 und 50 durchgeführt. Hier wurden die Kurven auch für 2% und 5% evaluiert, da es vermutet wurde, das gerade in dem Anfangsbereich der Kurve die Unterschiede am größten sein würden (da man dann noch wenig etikettierten Objekte hat und von der Information zur Verteilung profitieren sollte). Das Fazit war leider, dass es keinen messbaren Zugewinn an Performanz zwischen den zwei Möglichkeiten der Support-Messung gibt und dass die Kombination Support+ χ^2 überhaupt nicht besser ist als χ^2 alleine. Die Experimente wurden

Ergebniskurve 5 Lernkurve für den FONTAINE-Datensatz, verschiedene Merkmalsuchverfahren. Die Zahlen in Klammern entsprechen den Werten des Balanzierkoeffizienten β . 5-fache 10-Fold CV, Prozente bedeuten Anteil des Folds. Für Wrapper wurde eine Untermenge von FONTAINE verwendet (50% der Graphen zufällig ausgewählt, 50% der Merkmale nach χ^2). Sonstige Verfahren wurden auf dem ganzen Datensatz bewertet.



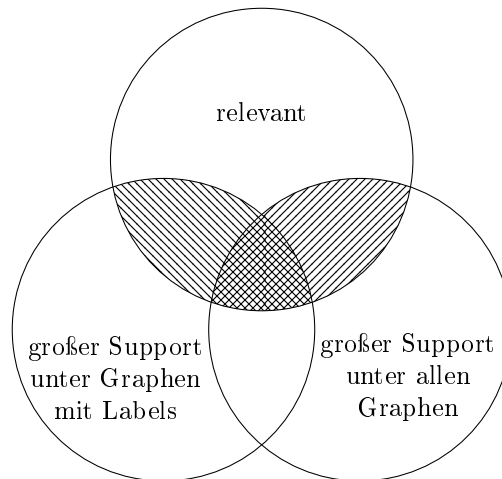
Ergebniskurve 6 Lernkurve für den FONTAINE-Datensatz, verschiedene Merkmalsuchverfahren. Die Zahlen in Klammern entsprechen den Werten des Balanzierkoeffizienten β . Hm. bedeutet den Hamming-Abstand, „alle“ bedeutet, dass alle Merkmale als Nachbarn berücksichtigt wurden. Für stochastische Suche wurde nur der Hamming-Abstand verwendet. 5-fache 10-Fold CV, Prozente bedeuten Anteil des Folds.



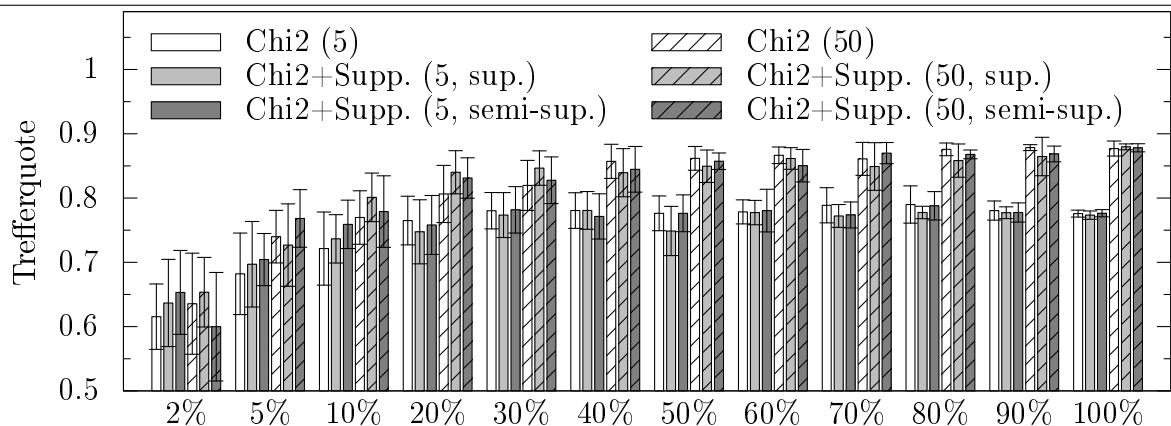
auch für andere Werte von k wiederholt, da allerdings auch hier keine Verbesserung der Trefferquote feststellbar war, werden sie in dieser Arbeit nicht angeführt.

3.6 Interpretation der Ergebnisse - Regression

Als Erstes wurde untersucht, wie gut verschiedene Regressionsverfahren mit den untersuchten Datensätzen funktionieren. Die Lernkurve 8 zeigt das Verhalten einer nu-SVM mit den EDKB-, BERGSTROM- und PDGFR-Datensätzen. Man sieht sofort, das bei allen drei ein Aufstieg der Kurve, also das Vorhandensein von Lernpotenzial, zu beobachten ist, wobei das Verhalten für den EDKB-Datensatz mit Abstand das beste ist. In den Lernkurven 23 und 24 sind Ergebnisse für jeweils EPA und COX zu sehen. Da für beide Datensätze die Resultate bei normaler Auswertung ziemlich mager waren, wurde untersucht, inwiefern die schlechten Werte von der Verteilung der Etiketten abhängen und inwiefern sie auf das tatsächliche Fehlen von Verallgemeinerungspotential hindeuten. Hierfür wurde eine Transformation konstruiert, die die Etiketten so umwandelt, dass das Histogramm gleichmäßig wird (d.h. eine Vertei-



Ergebniskurve 7 Lernkurve für den FONTAINE-Datensatz, Chi-2 mit und ohne Support-schranke (0.1). Die Zahl in Klammern ist die Zahl der verbleibenden Merkmale. 10-fache 10-Fold CV.

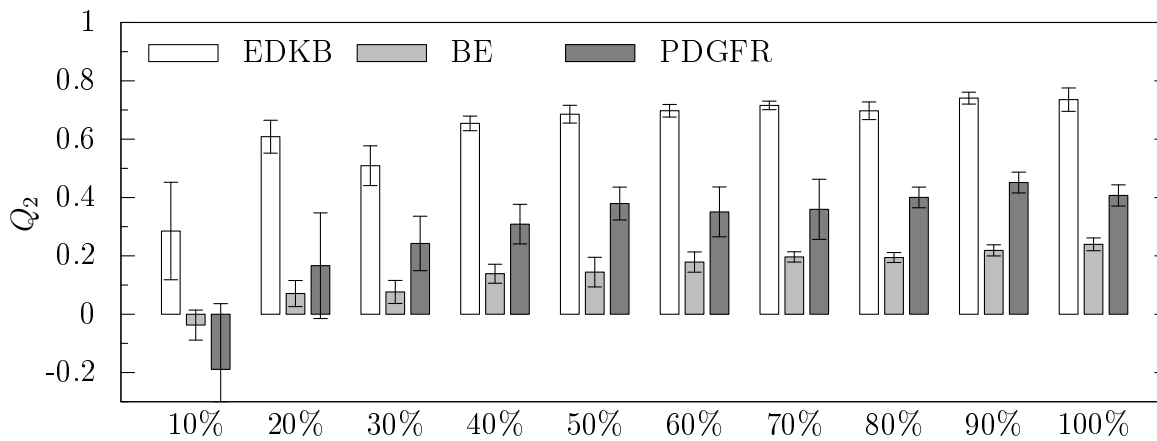


lungsfunktion, die die Verteilung approximiert¹³). Dann wurde mit den so transformierten Etiketten trainiert. Die Score-Funktion wurde dann auf zwei unterschiedliche Weisen berechnet – einmal mit den transformierten Etiketten und einmal mit den ursprünglichen (d.h die Schätzungen für die Etiketten wurden mit F^{-1} zurücktransformiert). Wie aus 23 und 24 ergeht, sind die Ergebnisse im ersteren Fall viel besser. Das war auch nicht anders zu erwarten, da die intern benutzte nu-SVM von der geänderten Verteilung nichts weiß und sie auch in ihrer Kostenfunktion nicht berücksichtigen kann. Beim EPA-Datensatz sind die so erzeugten Ergebnisse sehr schlecht, bei COX sind sie vergleichbar mit denen, wo mit der ursprünglichen Verteilung trainiert wurde. Die Ergebnisse, wo man mit den transformierten Etiketten testet sind dagegen in beiden Fällen sehr gut. Das bedeutet, dass auch in scheinbar schwierigen Datensätzen, wie COX oder EPA Lernpotenzial vorhanden ist – ansonsten lägen die Q_2 -Werte ja nah an, oder unter Null, und das unabhängig von der benutzten Transformation. Dieses Potenzial kann aber wegen der ungünstigen Verteilung der Etiketten¹⁴ nicht verwirklicht werden. Da dieses Auswertungsverfahren keine Standardtechnik ist, wurde in

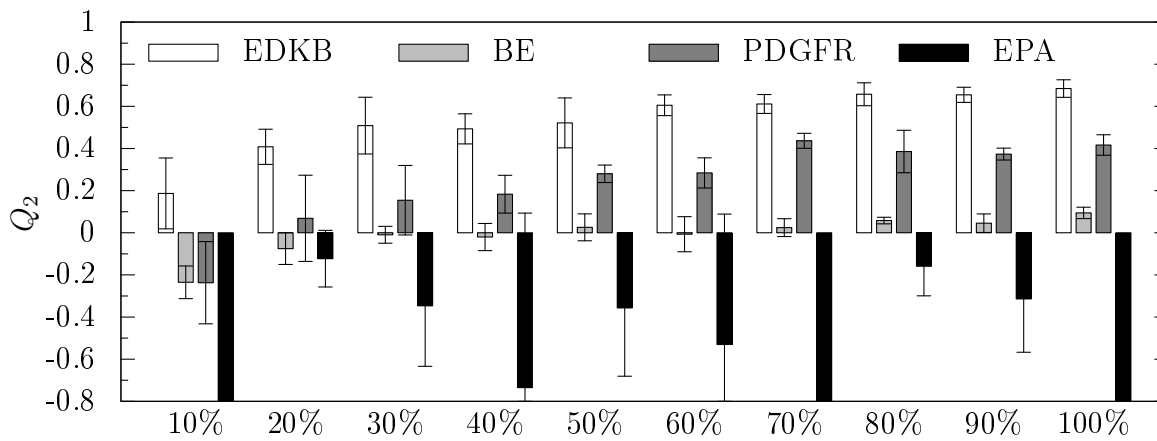
¹³Dieses wurde manuell in MATLAB gemacht.

¹⁴Man kann es auch als Mangel der herkömmlichen Regression-Performanzmaße wie MSE , MAE oder Q_2 sehen.

Ergebniskurve 8 Lernkurven für EDKB, BERGSTROM und PDGFR (nu-SVM). 5-fache 5-Fold CV, Prozente bedeuten Anteil des Folds.



Ergebniskurve 9 Lernkurven für EDKB, BERGSTROM, EPA und PDGFR (kNN, $k = 5$). 5-fache 5-Fold CV, Prozente bedeuten Anteil des Folds.



den nachfolgenden Abschnitten wieder nur direkte Auswertung angewandt.

Die Lernkurven 9 und 25 zeigen Ergebnisse, die mit dem kNN-Regressionsverfahren mit k von jeweils 5 und 50 erreicht wurden (wobei in den Anfangsteilen der Kurve natürlich nicht mehr Nachbarn benutzt werden, als in der Trainingsmenge vorhanden sind). Die Ergebnisse für die EDKB und PDGFR-Datensätze sind sich sehr ähnlich, wobei diejenigen für den kleineren Wert von k sogar um eine Nuance besser sind. Bei dem BERGSTROM-Datensatz ist Verallgemeinerungspotenzial ebenfalls feststellbar, wenn auch nur im begrenzten Maße. Für den EPA-Datensatz sind die Ergebnisse in beiden Fällen sehr schlecht – bei negativen Werten macht es wenig Unterschied wie sehr negativ sie sind. Es kann aber gesagt werden, dass die Ergebnisse für $k = 50$ wesentlich stabiler sind. Die Kurven 26 und 27 stellen Ergebnisse dar, die mit den beiden Regularisierungsverfahren erreicht wurden (siehe Abschnitte jeweils 2.4.2 und 2.4.3). Für EDKB und PDGFR waren beide Verfahren nah an den Ergebnissen, die mit nu-SVM machbar sind. Dies ist einerseits gut, da insbesondere direkte Regularisierung ein im Vergleich zu SVM einfach zu implementierendes Verfahren ist, andererseits enttäuschend, da bei einem Verfahren, das von vornherein transduktiv ist, also mehr Information zur Verfügung hat als eine SVM, eigentlich eine Verbesserung der Ergebnisse fällig wäre. Für den BERGSTROM-Datensatz ist direkte Regularisierung besser, es ist anders als bei Regularisierung mit Kernels ein kleiner Aufstieg der Kurve feststellbar. Für den EPA-Datensatz sind die Ergebnisse – ähnlich wie für andere Regressionsverfahren bei direkter Auswertung – schlecht, wobei die Kernel-Variante wesentlich stabiler ist.

Tabelle 3.7 Vergleich der Ergebnisse für Regression mit Daten aus Literatur.

Daten	Größe	Auswertung ^a	gemessen ^c	angegeben ^c	Quelle
COX	MAE	HO(100,90%)	0.72 (0.06)	1.02 (0.54)	Buchwald et al. [17], Tab. 2
EDKB/S	Q_2	CV(1,5)	0.74 (0.09)	0.65 (0.13)	Saigo et al. [69], Tab. 2
EDKB/S	Q_2	1-mal LOO	0.67	0.59	Saigo et al. [70], Tab. 3
BE	Q_2	CV(1,5) o.N. ^b	0.07 (0.02)	0.62 (0.02)	Ketkar et al. [45], Abb. 1b
BE	Q_2	CV(1,5) m.N. ^b	0.18 (0.16)		
PHEN	RMS	1-mal LOO	0.40	0.26	Yao et al. [95], Tab. 4

^a Bedeutung der Abkürzungen: CV(1,5) bedeutet einmalige 5-Fold-Kreuzvalidierung, LOO bedeutet Leave-one-out, HO(100,90%) bedeutet 100 Iterationen von Holdout, wobei mit 90% der Objekte trainiert wird.

^b Das Experiment wurde ohne Normierung der Etiketten (o.N.) und mit Normierung auf das Intervall (0,1) (m.N.) durchgeführt.

^c Die Zahlen in Klammern sind Standardabweichungen.

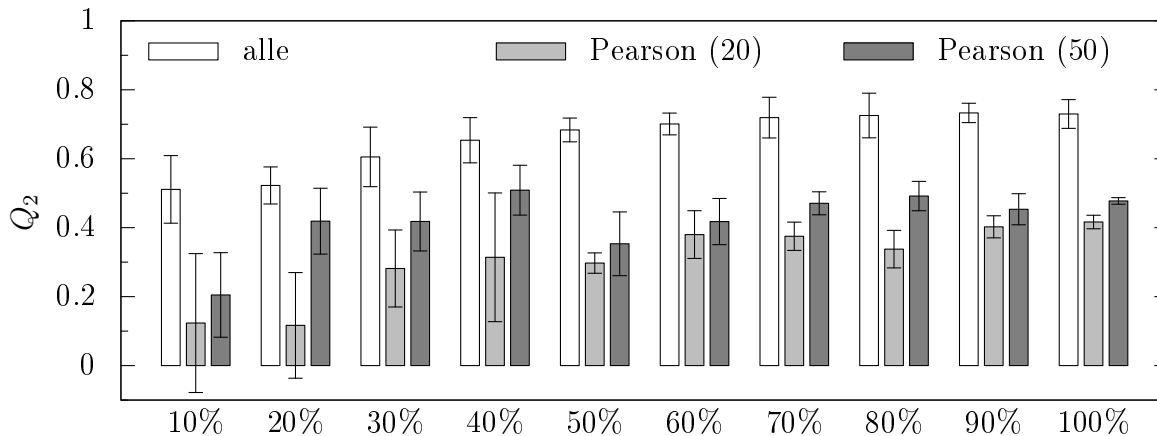
Damit die Pipeline auch für Regressionsprobleme validiert werden kann, wurde eine Reihe von Vergleichen mit Ergebnissen aus Literatur unternommen (siehe Tabelle 3.7). Es wurde hier keine Merkmalsuche verwendet, als Regressionsalgorithmus kam immer nu-SVM zum Einsatz. Die Auswertung wurde immer so durchgeführt, wie es im Vergleichspaper angegeben war. Es wurde, außer für den BERGSTROM-Datensatz, kein Versuch unternommen, die zitierten Experimente genau zu wiederholen – es wurde ja ein anderer Regressionsalgorithmus benutzt. Ferner war es das Ziel des Experiments, sicherzustellen, dass die erreichten Werte plausibel sind. Das ist angesichts des Vergleichs mit Buchwald et. al. [17] bestimmt der Fall: Der erreichte Wert war deutlich besser und das obwohl für die Berechnung in dieser Arbeit nur ein linearer Kernel benutzt wurde (quadratisch¹⁵ bei [17]). Die Ergebnisse für den EDKB/S-Datensatz sind besser¹⁶ als die Vergleichswerten in [69] und [70]. Dies zeugt davon, dass die Pipeline mit dem aktuellen Stand der Forschung gut korrespondiert. In diesem Kontext ist es verwunderlich, warum die Ergebnisse für den BERGSTROM-Datensatz aus [45] nicht reproduziert werden konnten, zumal das Auswertungsverfahren und die Implementierung von SVM genau die gleichen waren. Das Experiment wurde auch mit normierten Etiketten wiederholt, auch in dem Fall war das Ergebnis aber schlecht. Es wurde ein Versuch unternommen, bei den Autoren nachzufragen, ob sie eventuell von Zwischenschritten Gebrauch genommen haben, von denen im Paper nicht berichtet wurde; dieser blieb jedoch erfolglos. Schließlich wurde untersucht, wie gut die gewählte Methodologie (der Gebrauch von Untergraphen) im Vergleich mit chemischen Deskriptoren, wie sie von Zao et. al. [95] verwendet werden, abschneidet. Der für den PHENOL-Datensatz erzielte RMS Wert war schlechter als der Wert von Zao, blieb aber in der richtigen Größenordnung (es wird bei Zao zum Beispiel ein RMS-Ergebnis von 0.35 für Multiple linear Regression berichtet). Zusammenfassend kann man sagen: Die Ergebnisse für Regression, die in dieser Arbeit erzielt wurden sind trotz Probleme mit dem BERGSTROM-Datensatz glaubwürdig und bieten eine gute Basis, um Merkmalsuchverfahren zu testen.

Diese wurden als Nächstes untersucht. Es wurden hierfür die die EDKB- und PDGRF-Datensätze verwendet, da mit ihnen in den vorgängigen Abschnitten die besten Ergebnisse erzielt wurden. Die Kurven 10 und 28 stellen einen Vergleich dar zwischen den Ergebnissen, die mit allen Merkmalen und mit den ersten 20 beziehungsweise 50 nach dem Pearson-

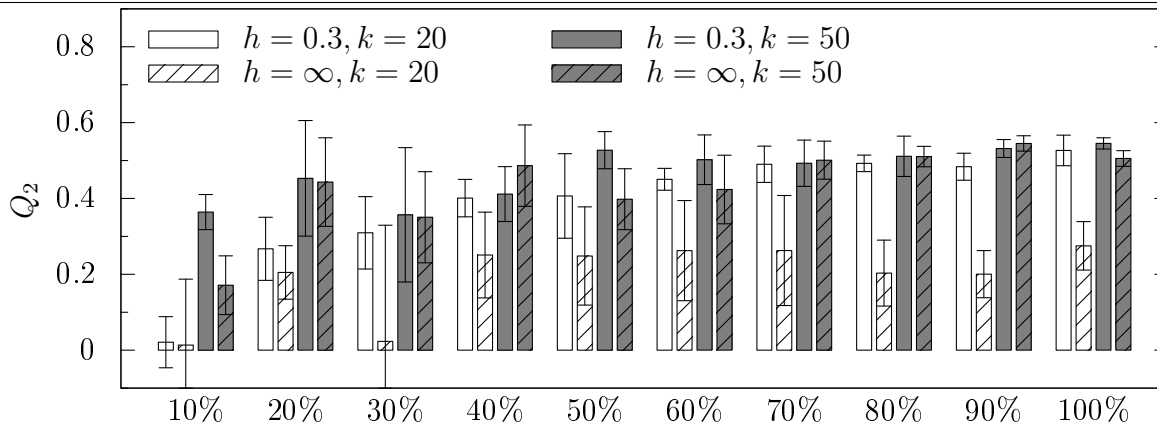
¹⁵Das Ergebnis von Buchwald et. al. für den RBF-Kernel wurde dagegen besser als der hier erreichte.

¹⁶Es wurde mit den SVM-Werten aus [70] verglichen, nicht mit Ergebnissen von gBoost.

Ergebniskurve 10 Vergleich verschiedener Merkmalsuchverfahren (nu-SVM, EDKB). Die Zahlen in Klammern entsprechen der Zahl verbleibender Attribute. 5-fache 5-Fold CV, Prozente bedeuten Anteil des Folds.



Ergebniskurve 11 Vergleich verschiedener Parameter für MI-Merkmalsuche (nu-SVM, EDKB). Hier ist h die Breite des Kerns, k die Anzahl der verbleibenden Merkmale. 5-fache 5-Fold CV, Prozente bedeuten Anteil des Folds.



Korrelationsscore (siehe 1.2.1) möglich sind. In beiden Fällen ist, wie zu erwarten wäre, eine Minderung des Q_2 -Scores bei Verkleinerung der Merkmalmenge zu beobachten. Die Ergebnisse für 50 Merkmale sind außer im instabilen Anfangsteil der Kurve für PDGFR immer besser als für 20. Für den EDKB-Datensatz sind Ergebnisse, die man mit Mutual-Information-Score erreicht (siehe Kurve 11) besser, als die mit Pearson-Score: Der Unterschied ist klein für 50 Merkmale aber deutlich für 20. Für den PDGFR-Datensatz (Kurve 29) werden die Unterschiede kleiner. Man kann aber trotzdem sagen, dass der MI-Score mindestens so gut funktioniert wie Pearson. Da ein wesentlicher Teil der Berechnung des MI-Scores die Approximation der Dichten $P(t)$, $P(t|\mathbf{f} = 1)$ und $P(t|\mathbf{f} = 0)$ ist, wurde untersucht, wie sehr sich dieser Schritt auf die Qualität des Scores auswirkt. Die schraffierten Balken in den Kurven 11 und 29 geben Q_2 -Werte an, wo die Dichte mit einer gleichmäßigen Funktion approximiert wurde (dies entspricht der unendlichen Kernelbreite). Die so errechneten Werte sind immer deutlich schlechter als die mit der optimalen Kernelbreite, wobei der Unterschied für $k = 20$ größer ist. Dies begründet die These, dass gute Dichtenapproximation für die Berechnung des MI-Scores kritisch ist und deutet darauf hin, dass bei Methoden der Kernelapproximation, die komplizierter sind als eine einfache Umsetzung von Parzen-Fenstern, wie sie hier verwendet wurde, eventuell weitere Zugewinne an Q_2 -Werten zu erwarten wären. Sie wurden in dieser Arbeit nicht untersucht.

Am ende wurde Merkmalsuche mit dem Rayleigh-Koeffizienten evaluiert. Die lineare Ver-

sion des Verfahrens lieferte sehr schlechte Ergebnisse (da alle Werte unter null lagen, wurde hier auf das Plotten einer Kurve verzichtet). Angesichts der schlechten Werte wurde manuell untersucht, ob in der verwendeten Eigenimplementierung die Matrizen C und N richtig generiert sind. Es wurde auch mittels eines Vergleichs mit MATLAB-Bibliotheksfunktion sichergestellt, dass das generalisierte Eigenwertproblem richtig gelöst wird. Am Ende kann man nur feststellen, dass der Ansatz für Probleme, wo die Anzahl der Merkmale die Anzahl der Trainingsobjekte deutlich überwiegt (was hier in dieser Arbeit der Fall ist) anscheinend nicht geeignet ist. Die schlechten Werte können auch dadurch verursacht sein, dass hier nur *ein* Hybridmerkmal generiert wurde (siehe Abschnitt 1.2.3 für Beschreibung, warum). Da die Ergebnisse mit dem linearen Ansatz schlecht waren und da bei SVMs der Umstieg auf nichtlineare Kernels keine Verbesserung brachte¹⁷, wurde auf die Implementierung der Kernel-Version verzichtet.

¹⁷Die entsprechenden Kurven wurden nicht geplottet, da es nicht im Fokus dieser Arbeit war.

Kapitel 4

Schlussfolgerungen und Weiterentwicklung

Der Zweck dieser Arbeit war dreifaltig: Erstens sollten die Probleme der Klassifizierung und Regression mittels Untergraphen untersucht, zweitens aus anderen Zweigen des Maschinellen Lernens bekannte Verfahren zur Merkmalsuche ausprobiert und drittens das Problem auf die Nützlichkeit von unetikettierten Graphen untersucht werden. Zeitbedingt konnte in jeder dieser Kategorien nur ein Teil der verfügbaren Methodik eingesetzt werden, es ist aber trotzdem möglich gewesen, zu wichtigen Einsichten zu erlangen. Zum einen wurde festgestellt, dass sowohl im Falle der Klassifizierung als auch Regression die von dem dem Autor untersuchten Algorithmen meistens nur für bestimmte Datensätze funktionieren. Das ist — unabhängig von den Werten von Auswertungsmaßen im Fall, wo sie gute Ergebnisse liefern — ein schlechtes Zeichen, da es eine der Grundannahmen des maschinellen Lernens und im breiteren Sinne der Wissenschaft überhaupt ist, dass sich eine Familie von Problemen mit einer gemeinsamen Theorie erklären lässt (also hier mit dem induktiven Bias einer Klassifizierungs- oder Regressionspipeline). Das Problem wird dadurch bestärkt, dass in den meisten Veröffentlichungen, die z.B. die neue Variante eines Klassifizierers oder ein Merkmalsuchverfahren einführen, nur positive Ergebnisse berichtet werden, die Experimente mit ungünstigen Ausgängen werden ausgeschaltet. Dadurch entsteht ein verzerrtes Bild vom Stand der Forschung: Die Methoden sind nicht so gut, wie sie verkauft werden. Sie können nur dann zuverlässig evaluiert werden, wenn man zumindest Teile davon selber implementiert hat, was zeitaufwändig ist. In dieser Arbeit wurden alle Ergebnisse berichtet, nicht nur die positiven. Was die Merkmalsuche angeht, wurde sowohl für Klassifizierung als auch für Regression festgestellt, dass die einfachsten Ansätze entweder am besten funktionieren oder die Ergebnisse, die man mit ihnen erreichen kann, sich zu wenig vom Optimum unterscheiden, als das der vermehrte Aufwand einer komplizierten Methode sinnvoll wäre: Die Baseline-Verfahren (χ^2 und Pearson) schnitten überdurchschnittlich gut aus; ein Blick auf zum Beispiel Kurve 5 genügt, um Diskussionen, welches Verfahren besser ist, Haarspalterei ähneln zu lassen. Die Versuche, unetikettierte Graphen zur Verbesserung der Ergebnisse zu verwenden sind nicht geglückt: Sowohl in der Phase der Merkmalsuche (skalarproduktbasierter Score bei Klassifizierung und Rayleigh-Koeffizient bei Regression) als auch in der eigentlichen Klassifizierung (TSVMs bei Klassifizierung und Regularisierungsverfahren bei Regression) brachte das Wissen über die zusätzlichen Moleküle keine Verbesserung. Trotz allem Vorgängigen erscheint es positiv, dass diese Arbeit im Lichte der durchgeführten Literaturvergleiche nicht schlechter ist, als die anderen.

Da das Problem des Lernens aus chemischen Molekülen noch relativ jung ist, sind in praktisch allen Schritten der Pipeline in der Zukunft Verbesserungen zu erwarten. Für den Schritt der Extraktion wäre es nützlich, einen Algorithmus zu konstruieren, der auf den

theoretischen Erkenntnissen zur Charakterisierung der chemischen Graphen basiert, wie zum Beispiel auf beschränkter Baumweite und der für Graphen, die diese Charakterisierung erfüllen, polynomiell ist. In der Merkmalsuche wäre es nützlich sich trotz der hier berichteten schlechten Ergebnissen mit der Version des Rayleigh-Koeffizienten von Pan et al. [65] andere Varianten davon anzuschauen — dieser Ansatz ist sehr allgemein, da vieles von der richtigen Konstruktion der zwei verwendeten Matrizen abhängig ist. Ebenfalls interessant wäre eine Erweiterung der ICA-Idee, wie sie von Sejnowski et al. [5] eingeführt und von Kwak et al. [50] um Korrelationen mit der Zielvariable erweitert wurde. Da hier das Basisverfahren iterativ und seit Jahren gut erforscht ist, kann mit kurzen Laufzeiten gerechnet werden. Sowohl ICA als auch Varianten vom Rayleigh-Koeffizienten sind in der Statistik etabliert und finden breite Verwendung — es müsste eigentlich möglich sein, sie auf Molekülklassifikation oder -regression anzupassen. In dem Bereich der Klassifizierer ist nach Ansicht des Autors nicht davon auszugehen, dass sich TSVMs als das ultimative Transduktionsverfahren behaupten. Das hauptsächliche Problem besteht darin, dass sie nicht selbstkonsistent¹ sind und, was damit verbunden ist, die generierte Entscheidungsregel nicht unbedingt optimal ist. Es ist also zu erwarten, dass entweder Hybridmodelle, die aus einer SVM und einem Graphenmodell der Abstände zwischen den Objekten bestehen, die Rolle des führenden transduktiven Verfahrens übernehmen oder es muss ein ganz neuer theoretisch gut fundierter Ansatz auftauchen. Auch bei transduktiver Regression ist definitiv nicht das letzte Wort gesagt worden: Es gibt zwar viele Verfahren, doch sie haben außer der äußerst vagen Idee der Regularisierung, die großzügig interpretiert werden kann wenig theoretische Fundierung. Bei der Auswertung von Regressionsverfahren müsste man schließlich eine Methode finden, die einerseits die Verteilung der Etiketten miteinbezieht und andererseits plausibel zu begründen ist. Wir haben gesehen, dass die Etikettenverteilung Qualitätsmaße wie Q_2 sehr stark beeinflusst. Es gibt zum Beispiel in dem EPA-Datensatz 74 Etiketten, die kleiner gleich eins sind und 85, die größer als 400 sind. Man sollte sich vor der eigentlichen Auswertung die Fragen stellen, ob die kleinen Etiketten nicht etwa wie ein Cluster behandelt werden sollten, ob solch kleine Unterschiede überhaupt außerhalb des Messungsfehlers sind sowie ob es wirklich einen Unterschied zwischen großen Werten wie 4510 und 653 gibt, wo ein klassisch gemessener Fehler äußerst groß werden kann. Mit einem Pauschalscore wie MSE ist das nicht möglich. Es wäre deshalb ratsam, zu untersuchen, was für eine Größe wirklich gemessen werden sollte. Dies könnte dann zur Konstruktion eines Besseren Regressionsverfahrens benutzt werden — da man die neue Größe optimieren könnte.

All die vorher erwähnten Sachen stellen inkrementelle Verbesserungen dar. Was aber wirklich einen spürbaren Fortschritt nach sich zöge, wäre das Umdenken der verwendeten Transformation der Deskriptorenmenge, also etwa der Untergraphen in den Raum, wo Klassifizierung oder Regression stattfindet. Die meist verwendeten Klassifizierer (SVMs mit polynomiellen oder Gausschen Kernels) operieren auf \mathbb{R}^n oder einer Euklidischen Erweiterung davon. Diese Räume haben Eigenschaften (zum Beispiel sind sie dicht), die Molekülsätze naturbedingt nicht haben. Die zentrale Frage zur langfristigen Anwendbarkeit von SVMs für Molekülklassifizierung ist die Frage nach der Einbettbarkeit molekularer Eigenschaften in Hilberträume. Es kann sein, dass der Begriff des Skalarprodukts ausreicht, um sie angemessen zu modellieren (einen Versuch in diese Richtung stellen 3D-Kernels dar). Es kann aber auch sein, dass ein reicherer Formalismus benötigt wird.

¹Die generierte Zuweisung der Etiketten und die generierte Entscheidungsregel sind nicht kompatibel, das heißt es kann sein das ein Molekül ein anderes Etikett zugewiesen bekommt als sich aus der Entscheidungsregel ergibt.

Literaturverzeichnis

- [1] David W. Aha, Dennis Kibler, and Marc K. Albert. Instance-based learning algorithms. *Machine learning*, 6(1):37–66, 1991.
- [2] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k -tree. *SIAM Journal on Algebraic and Discrete Methods*, 8(2):277–284, 1987.
- [3] Roberto Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE transactions on neural networks*, 5(4):537–550, 1994.
- [4] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *The Journal of Machine Learning Research*, 7:2434, 2006.
- [5] Anthony J. Bell and Terrence J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, 7(6):1129–1159, 1995.
- [6] Yoshua Bengio and Yves Grandvalet. No unbiased estimator of the variance of k -fold cross-validation. *The Journal of Machine Learning Research*, 5:1105, 2004.
- [7] Kristin P. Bennett and Ayhan Demiriz. Semi-supervised support vector machines. In *Advances in neural information processing systems 11: proceedings of the 1998 conference*, page 368. The MIT Press, 1999.
- [8] Christel A. S. Bergström, Ulf Norinder, Kristina Luthman, and Per Artursson. Molecular descriptors influencing melting point and their role in classification of solid drugs. *J. Chem. Inf. Comput. Sci.*, 43(4):1177–1185, 2003.
- [9] Avrim Blum, Adam Kalai, and John Langford. Beating the hold-out: Bounds for k -fold and progressive cross-validation. In *Proceedings of the twelfth annual conference on Computational learning theory*, page 208. ACM, 1999.
- [10] Avrim L. Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial intelligence*, 97(1-2):245–271, 1997.
- [11] Brian V. Bonnländer and Andreas S. Weigend. Selecting input variables using mutual information and nonparametric density estimation. In *Proceedings of the 1994 International Symposium on Artificial Neural Networks*, pages 42–50, 1994.
- [12] Christian Borgelt, Michael R. Berthold, and David E. Patterson. Molecular fragment mining for drug discovery. *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, 3571/2005:1002–1013, 2005.
- [13] Karsten Borgwardt, Xifeng Yan, Marisa Thomas, Hong Cheng, Arthur Gretton, Le Song, Alex Smola, Jiawei Han, Philip Yu, and Hans Peter Kriegel. Combining near-optimal feature selection with gSpan. *Mining and Learning on Graphs Workshop (MLG 2008)*, Helsinki, 2008.
- [14] Endre Borosa, Takashi Horiyama, Toshihide Ibaraki, Kazuhisa Makino, and Mutsunori Yagiura. Finding essential attributes from binary data. *Annals of Mathematics and Artificial Intelligence*, 39(3):223–257, 2003.
- [15] Andrew P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.
- [16] Ulisses M. Braga-Neto and Edward R. Dougherty. Is cross-validation valid for small-sample microarray classification? *Bioinformatics*, 20(3):374, 2004.
- [17] Fabian Buchwald, Tobias Girschick, Madeleine Seeland, and Stefan Kramer. Using local models to improve (Q)SAR predictivity. To be published.
- [18] Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.
- [19] Olivier Chapelle, Mingmin Chi, and Alexander Zien. A continuation method for semi-supervised SVMs. In *Proceedings of the 23rd international conference on Machine learning*, page 192. ACM, 2006.
- [20] Olivier Chapelle, Vikas Sindhwani, and S. Sathya Keerthi. Branch and Bound for Semi-Supervised Support Vector Machines. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 217–224. MIT Press, Cambridge, MA, 2007.

- [21] Olivier Chapelle, Vikas Sindhwani, and Sathya S. Keerthi. Optimization techniques for semi-supervised support vector machines. *The Journal of Machine Learning Research*, 9:203–233, 2008.
- [22] Eduardo Bayro Corrochano. *Handbook of Geometric Computing: Applications in Pattern Recognition, Computer Vision, Neural Computing, and Robotics*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.
- [23] Corinna Cortes and Mehryar Mohri. On transductive regression. In *NIPS*, pages 305–312, 2006.
- [24] Corinna Cortes, Mehryar Mohri, Dmitry Pechyony, and Ashish Rastogi. Stability of transductive regression algorithms. In *Proceedings of the 25th international conference on Machine learning*, pages 176–183. ACM, 2008.
- [25] Thomas M. Cover. Rates of convergence for nearest neighbor procedures. <http://www.stanford.edu/~cover/papers/paper009.pdf>, 1968.
- [26] Thomas M. Cover and Peter E. Hart. Nearest Neighbor Pattern Classification. *IEEE Trans. on Information Theory*, 13(1):pp 21–27, January 1967.
- [27] Manoranjan Dash and Huan Liu. Feature Selection for Classification. *Intelligent Data Analysis*, 1:131–156, 1997.
- [28] Manoranjan Dasha, Huan Liu, and Hiroshi Motoda. Consistency based feature selection. *Knowledge Discovery and Data Mining. Current Issues and New Applications*, 1805:98–109, 2000.
- [29] Morris H. DeGroot and Mark J. Schervish. *Probability and statistics*. Addison Wesley Longman, 2002.
- [30] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000.
- [31] David Eppstein. Subgraph isomorphism in planar graphs and related problems. In *SODA '95: Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 632–640, Philadelphia, PA, USA, 1995. Society for Industrial and Applied Mathematics.
- [32] Tom Fawcett. An introduction to ROC analysis. *Pattern recognition letters*, 27(8):861–874, 2006.
- [33] George Fix and Richard Heiberger. An algorithm for the ill-conditioned generalized eigenvalue problem. *SIAM Journal on Numerical Analysis*, 9(1):78–88, 1972.
- [34] Fabien Fontainea, Manuel Pastor, Ismael Zamora, and Ferran Sanz. Anchor-GRIND: Filling the Gap between Standard 3D QSAR and the GRid-INdependent Descriptors. *J. Med. Chem.*, 48(7):2687–2694, 2005.
- [35] Johannes Fürnkranz. Separate-and-Conquer Rule Learning. *Artificial Intelligence Review*, 13(1):3–54, 1999.
- [36] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, January 1979.
- [37] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.
- [38] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.
- [39] John E. Hopcroft and Jin K. Wong. Linear time algorithm for isomorphism of planar graphs (preliminary report). In *STOC '74: Proceedings of the sixth annual ACM symposium on Theory of computing*, pages 172–184. ACM, New York, NY, USA, 1974.
- [40] Tamás Horváth and Jan Ramon. Efficient frequent connected subgraph mining in graphs of bounded tree-width. *Theoretical Computer Science*, 411(31-33):2784 – 2797, 2010.
- [41] Piotr Indyk and Rameev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM New York, NY, USA, 1998.
- [42] Thorsten Joachims. Transductive Inference for Text Classification using Support Vector Machines. In *Machine learning: proceedings of the Sixteenth International Conference (ICML'99), Bled, Slovenia, June 27-30, 1999*, page 200. Morgan Kaufmann Pub, 1999.
- [43] Michael I. Jordan. Lecture notes for advanced topics in learning and decision making course. <http://www.cs.berkeley.edu/~jordan/courses/281B-spring04/>, 2004.
- [44] Matti Kääriäinen. Semi-supervised model selection based on cross-validation. In *IJCNN'06. International Joint Conference on Neural Networks*, pages 1894–1899, 2006.
- [45] Nikhil S. Ketkar, Lawrence B. Holder, and Diane J. Cook. gRegress: extracting features from graph transactions for regression. In *Proceedings of the 21st international joint conference on Artificial intelligence*, pages 1089–1094. Morgan Kaufmann Publishers Inc., 2009.

- [46] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International joint Conference on artificial intelligence*, volume 14, pages 1137–1145. Citeseer, 1995.
- [47] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2):273–324, 1997.
- [48] Ron Kohavi and J. Ross Quinlan. *Data mining tasks and methods: Classification: decision-tree discovery*, pages 267–276. Oxford University Press, Inc., New York, NY, USA, 2002.
- [49] Daphne Koller and Mehran Sahami. Toward Optimal Feature Selection. In *Machine learning: proceedings of the Thirteenth International Conference (ICML'96)*, page 284. Morgan Kaufmann Pub, 1996.
- [50] Nojun Kwak and Chong-Ho Choi. Feature extraction based on ica for binary classification problems. *IEEE Transactions on Knowledge and Data Engineering*, 15 no. 6:1374–1388, 2003.
- [51] Pat Langley and Wayne Iba. Average-case analysis of a nearest neighbor algorithm. In *Proceedings of the 13th international joint conference on Artificial intelligence-Volume 2*, pages 889–894. Morgan Kaufmann Publishers Inc., 1993.
- [52] Alan J. Laub. *Matrix analysis for scientists and engineers*. Society for Industrial Mathematics, 2005.
- [53] Hu Li, Chun Wei Yap, Choong Yong Ung, Ying Xue, Zhi Wei Cao, and Yu Zong Chen. Effect of Selection of Molecular Descriptors on the Prediction of Blood- Brain Barrier Penetrating and Nonpenetrating Agents by Statistical Learning Methods. *J. Chem. Inf. Model*, 45(5):1376–1384, 2005.
- [54] Eugene M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, 25(1):42–65, 1982.
- [55] Jiří Matoušek and Robin Thomas. On the complexity of finding iso-and other morphisms for partial k-trees. *Discrete mathematics*, 108(1-3):343–364, 1992.
- [56] Andreas Maunz, Christoph Helma, Tobias Cramer, and Stefan Kramer. Latent structure pattern mining. Not published yet, prepared for ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2010.
- [57] Thorsten Meinl, Christian Borgelt, and Michael R. Berthold. Discriminative Closed Fragment Mining and Perfect Extensions in MoFa. In *Proc. 2nd Starting AI Researchers' Symposium (STAIRS 2004, Valencia, Spain)*, pages 3–14, 2004.
- [58] Sebastian Mika, Gunnar Rätsch, Jason Weston, Bernhard Schölkopf, Alex Smola, and Klaus-Robert Müller. Constructing descriptive and discriminative nonlinear features: Rayleigh coefficients in kernel feature spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25 no. 5:623–633, 2003.
- [59] Cleve B. Moler and Pete Stewart. An algorithm for generalized matrix eigenvalue problems. *SIAM Journal on Numerical Analysis*, 10(2):241–256, 1973.
- [60] Shinichi Morishita and Jun Sese. Transversing itemset lattices with statistical metric pruning. In *Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 226–236. ACM, 2000.
- [61] George L. Nemhauser, Laurence A. Wolsey, and Marschall L. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14(1):265–294, 1978.
- [62] John C. Platt. *Probabilistic outputs for support vector machines and comparison to regularized likelihood methods*, chapter 5, pages 61–74. Cambridge, MA: MIT Press, 1999.
- [63] Foster Provost and Tom Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42(3):203–231, 2001.
- [64] Foster Provost, Tom Fawcett, and Ron Kohavi. The Case against Accuracy Estimation for Comparing Induction Algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 445–453. Morgan Kaufmann Publishers Inc., 1998.
- [65] Qiang Yang Rong Pan and Lei Li. Case retrieval using nonlinear feature-space transformation. *Advances in Case-Based Reasoning*, 3155/2004:75–87, 2004.
- [66] Christine L. Russom, Steven P. Bradbury, Steven J. Broderius, Dean E. Hammermeister, and Robert A. Drummond. Predicting modes of toxic action from chemical structure: Acute toxicity in the fathead minnow (*Pimephales promelas*). *Environmental Toxicology and Chemistry*, 16(5):948–967, 1997.
- [67] Christoph Rücker and Markus Meringer. How many organic compounds are graph-theoretically nonplanar? *Communications in Mathematical and in Computer Chemistry/MATCH*, 45:153–172, 2002.
- [68] Ulrich Rückert and Stefan Kramer. Optimizing feature sets for structured data. *Machine Learning: ECML 2007*, 4701:716–723, 2007.

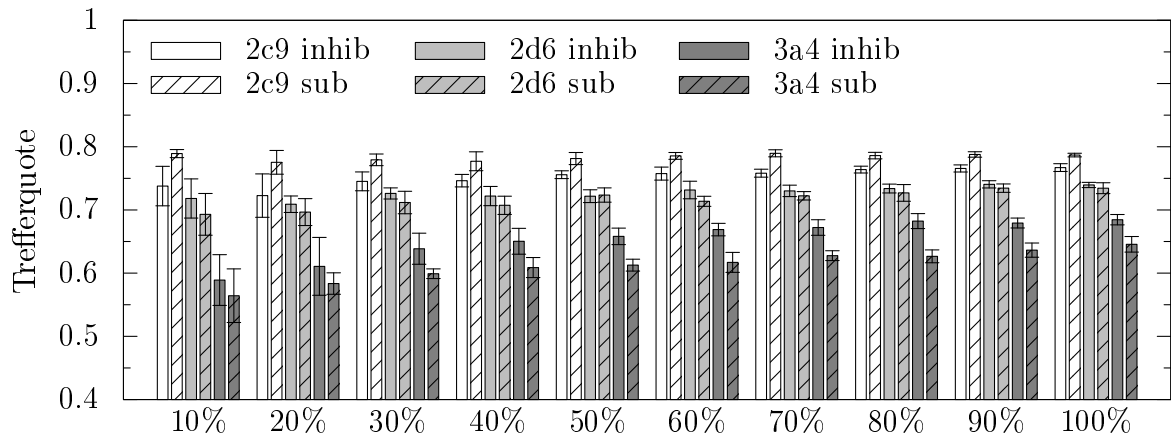
- [69] Hiroto Saigo, Nicole Krämer, and Koji Tsuda. Partial least squares regression for graph mining. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 578–586. ACM, 2008.
- [70] Hiroto Saigo, Sebastian Nowozin, Tadashi Kadowaki, Taku Kudo, and Koji Tsuda. gBoost: a mathematical programming approach to graph classification and regression. *Machine learning*, 75(1):69–89, 2009.
- [71] Steven L. Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*, 1(3):317–328, 1997.
- [72] Jean Pierre Sauvage. Interlacing molecular threads on transition metals: Catenands, catenates, and knots. *Accounts of Chemical Research*, 23(10):319–327, 1990.
- [73] Dale Schuurmans and Finnegan Southey. Metric-based methods for adaptive model selection and regularization. *Machine Learning*, 48(1):51–84, 2002.
- [74] Jon Shlens. A Tutorial on Principal Component Analysis. <http://www.sn1.salk.edu/~shlens/pca.pdf>.
- [75] Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.
- [76] Daniel A. Spielman. Lecture notes for spectral graph theory lecture. <http://www.cs.yale.edu/homes/spielman/561/>, 2009.
- [77] Ashwin Srinivasan and Ross D. King. Feature construction with inductive logic programming: A study of quantitative predictions of biological activity aided by structural attributes. *Data Mining and Knowledge Discovery*, 3(1):37–57, 1999.
- [78] Gilbert Strang. *Computational science and engineering*. Wellesley-Cambridge Press, 2007.
- [79] Jeffrey J. Sutherland, Lee A. O’Brien, and Donald F. Weaver. Spline-Fitting with a Genetic Algorithm: A Method for Developing Classification Structure- Activity Relationships. *J. Chem. Inf. Comput. Sci.*, 43(6):1906–1915, 2003.
- [80] Sebastian Thrun Suzanna Becker and Klaus Obermayer, editors. *Stability-based model selection*, volume 15. MIT Press, 2003.
- [81] S. Joshua Swamidass, Jonathan Chen, Jocelyne Bruand, Peter Phung, Liva Ralaivola, and Pierre Baldi. Kernels for small molecules and the prediction of mutagenicity, toxicity and anti-cancer activity. *Bioinformatics*, 21(Suppl 1):i359, 2005.
- [82] Martin Szummer and Tommi Jaakkola. Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems*, pages 945–952. MIT Press, 2002.
- [83] George R. Terrell and David W. Scott. Variable kernel density estimation. *The Annals of Statistics*, 20(3):1236–1265, 1992.
- [84] Marisa Thoma, Hong Cheng, Arthur Gretton, Jiawei Han, Hans Peter Kriegel, Alex Smola, Le Song, Philip S. Yu, Xifeng Yan, and Karsten Borgwardt. Near-optimal supervised feature selection among frequent subgraphs. In *In SIAM Int’l Conf. on Data Mining*, 2009.
- [85] Roberto Todeschini and Viviana Consonni. *Molecular Descriptors for Chemoinformatics*. Wiley-VCH Weinheim, Germany, 2009.
- [86] Weida Tong, Hong Fang, ClarLynda R. Williams, Jon M. Burch, and Ann M. Richard. DSSTox FDA National Center for Toxicological Research Estrogen Receptor Binding Database (NCTRER): SDF files and website documentation. http://www.epa.gov/nctt/dsstox/sdf_nctrer.html. Updated version 4b_232, 15 February 2008.
- [87] Kari Torkkola. Feature extraction by non parametric mutual information maximization. *The Journal of Machine Learning Research*, 3:1438, 2003.
- [88] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.
- [89] David H. Wolpert. On the Connection between In-sample Testing and Generalization Error. *Complex Systems*, 6:47–94, 1992.
- [90] Mingrui Wu and Bernhard Schölkopf. Transductive classification via local learning regularization. In X. Shen Meila, M., editor, *11th International Conference on Artificial Intelligence and Statistics*, pages 628–635, Brookline, MA, USA, 03 2007. Microtome.
- [91] Marc Wörlein, Thorsten Meinl, Ingrid Fischer, and Michael Philippsen. A quantitative comparison of the subgraph miners MoFa, gSpan, FFSSM, and Gaston. *Knowledge Discovery in Databases: PKDD 2005*, 3721/2005:392–403, 2005.
- [92] Atsuko Yamaguchi, Kiyoko F. Aoki, and Hiroshi Mamitsuka. Graph Complexity of Chemical Compounds in Biological Pathways. *Genome Informatics*, 14:376–377, 2003.
- [93] Xifeng Yan and Jiawei Han. gSpan: Graph-based substructure pattern mining, 2002.

- [94] Xifeng Yan and Jiawei Han. CloseGraph: mining closed frequent graph patterns. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, page 295. ACM, 2003.
- [95] XJ Yao, A. Panaye, JP Doucet, RS Zhang, HF Chen, MC Liu, ZD Hu, and BT Fan. Comparative study of QSAR/QSPR correlations using support vector machines, radial basis function neural networks, and multiple linear regression. *J. Chem. Inf. Comput. Sci*, 44(4):1257–1266, 2004.
- [96] Chen Yu-Zong and C.W. Yap. Prediction of cytochrome P450 3A4, 2D6, and 2C9 inhibitors and substrates by using support vector machines. *J. Chem. Inf. Model*, 45(4):982–992, 2005.
- [97] Mohammed Javeed Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, and Wei Li. New algorithms for fast discovery of association rules. In *In 3rd Intl. Conf. on Knowledge Discovery and Data Mining*, pages 283–286. AAAI Press, 1997.
- [98] Xiaojin Zhu, Zoubin Ghahramani, and John D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In Tom Fawcett, Nina Mishra, Tom Fawcett, and Nina Mishra, editors, *ICML*, pages 912–919. AAAI Press, 2003.

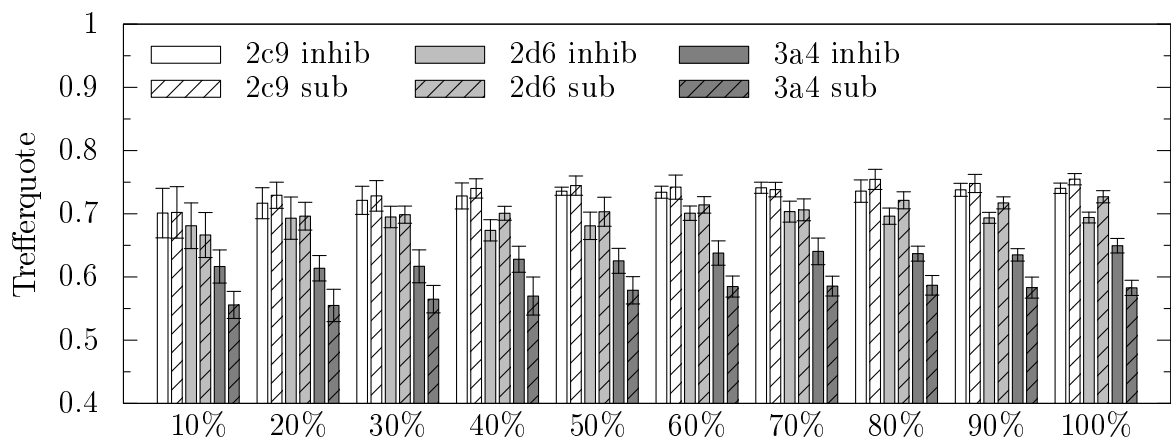
Anhang A

Ergebnisse - Klassifizierung

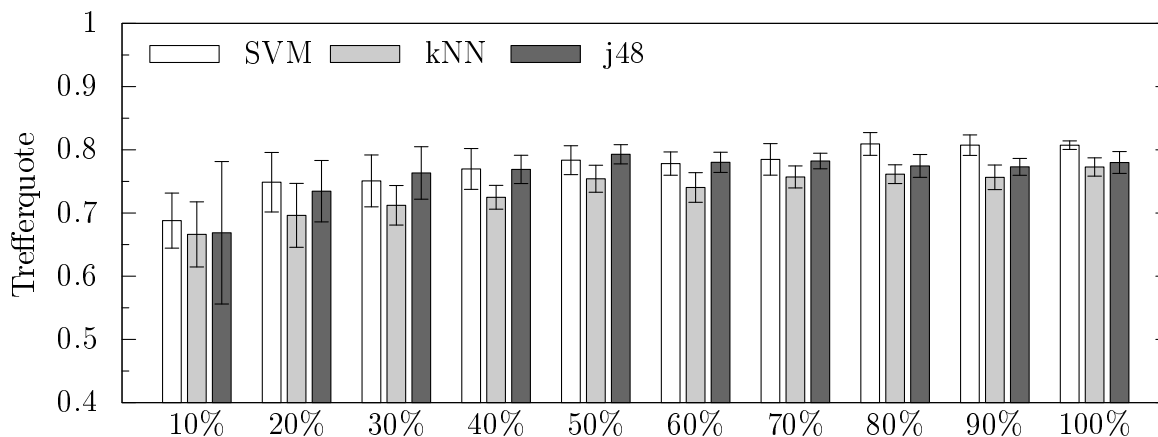
Ergebniskurve 12 Lernkurven für CYP-Datensätze (kNN). 10-fache 10-Fold CV, Prozenzte bedeuten Anteil des Folds.



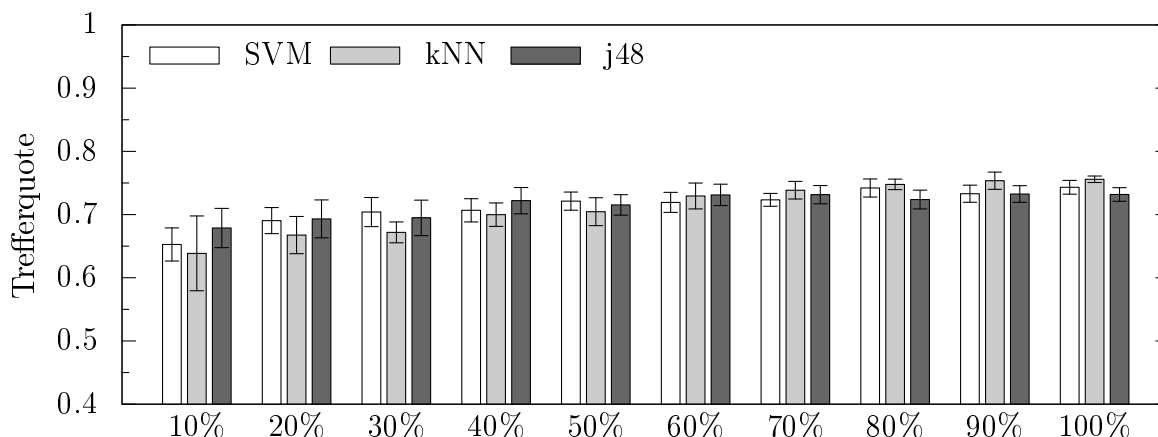
Ergebniskurve 13 Lernkurven für CYP-Datensätze (j48). 10-fache 10-Fold CV, Prozenzte bedeuten Anteil des Folds.



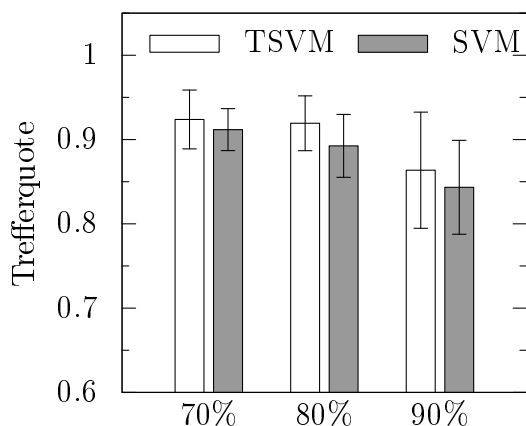
Ergebniskurve 14 Lernkurven für den NCTRER-Datensatz. 10-fache 10-Fold CV, Prozente bedeuten Anteil des Folds.



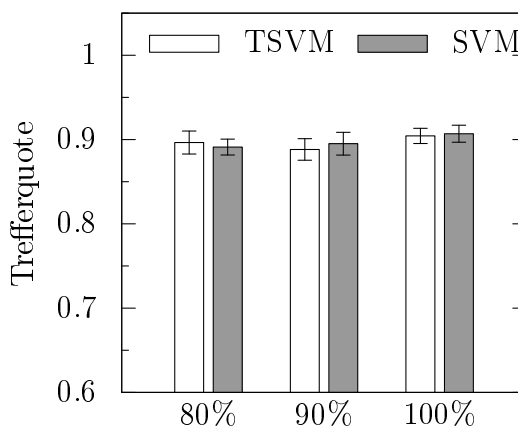
Ergebniskurve 15 Lernkurven für den BBP2-Datensatz. 10-fache 10-Fold CV, Prozente bedeuten Anteil des Folds.



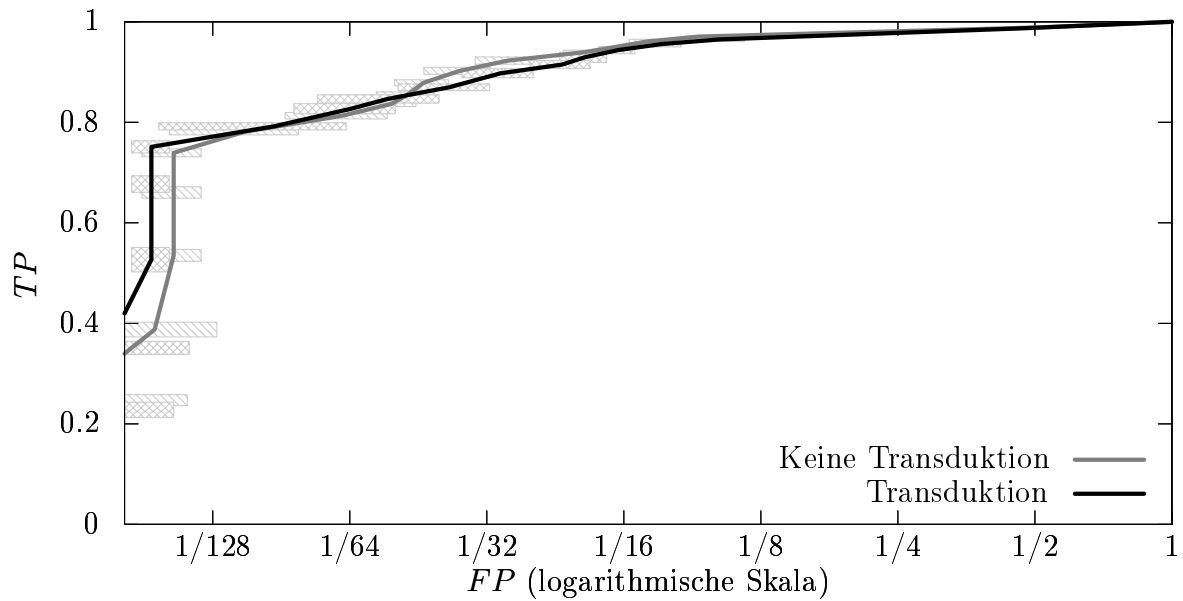
Ergebniskurve 16 Lernkurve: Untermenge von FONTAINE (50% der Graphen zufällig ausgewählt, 50% der Merkmale nach χ^2), TSVM, 20-faches Holdout. Prozente bedeuten Anteil des Datensatzes.



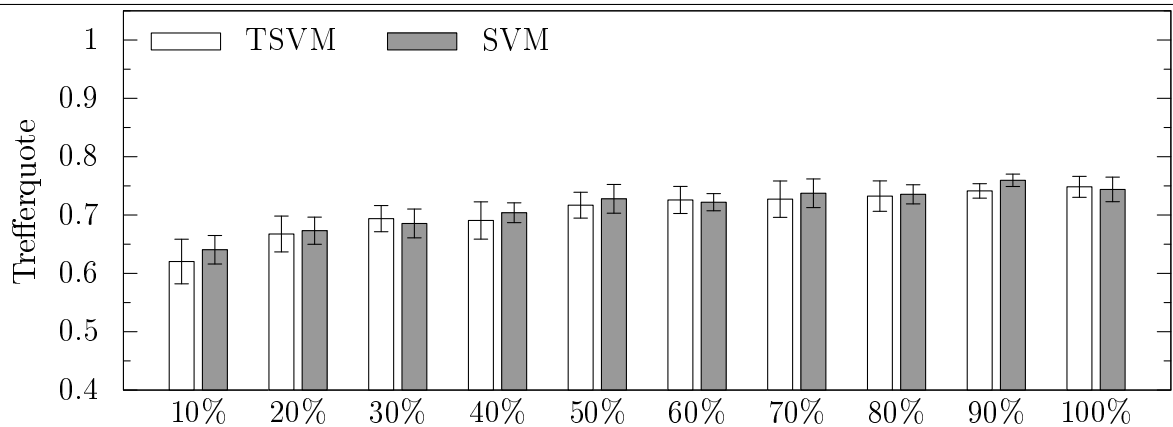
Ergebniskurve 17 Lernkurve: teilüberwacht (keine Transduktion). Untermenge von FONTAINE (30% der Graphen zufällig ausgewählt, 20% der Merkmale nach χ^2). 10-fache 10-Fold CV, Prozente bedeuten Anteil des Folds.



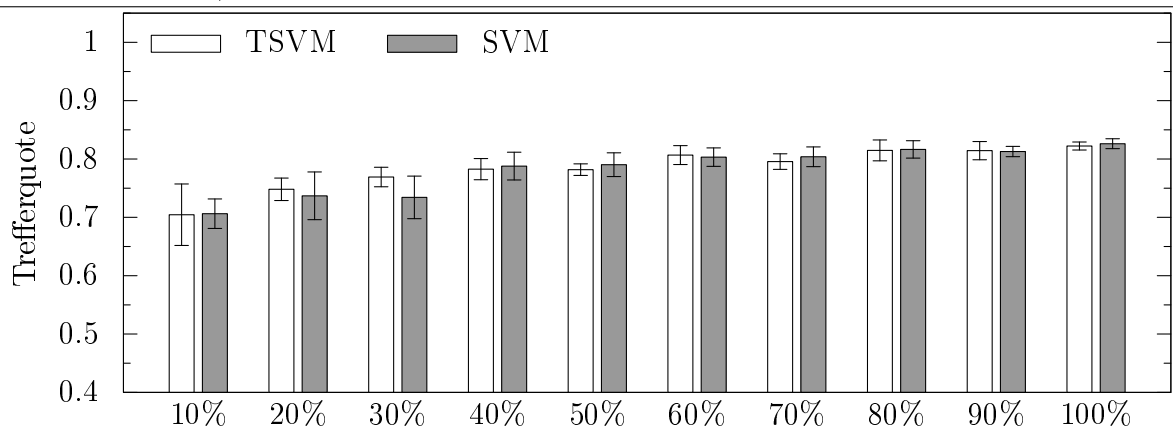
Ergebniskurve 18 ROC-Kurve: FONTAINE, TSVM vs. SVM (Training mit 100% des Folds, 10-fache 10-Fold CV). Die schattierten Bereiche entsprechen der Standardabweichung des (TP, FP) -Paares.



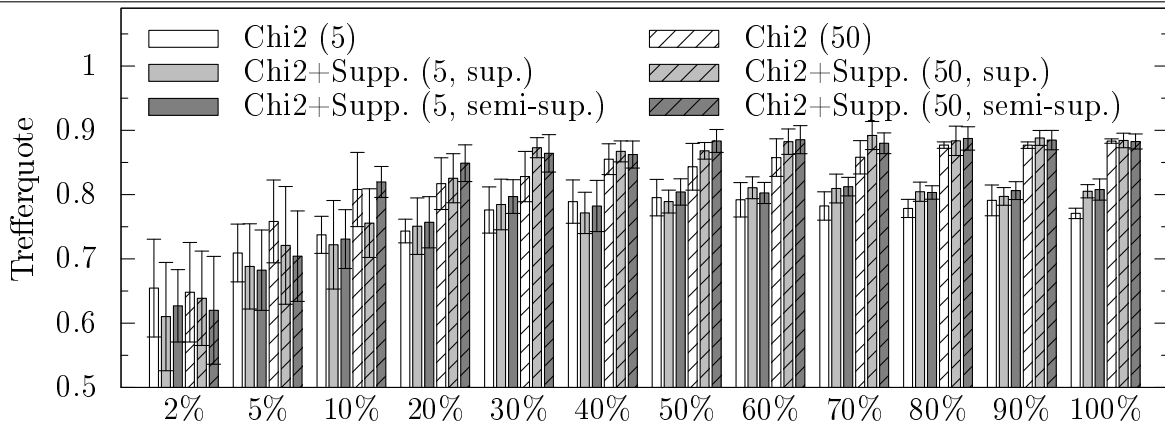
Ergebniskurve 19 Lernkurve, TSVM, Kreuzvalidierung. Untermenge von BBP2 (50% der Graphen zufällig ausgewählt, 50% der Merkmale nach χ^2) 10-fache 10-Fold CV, Prozente bedeuten Anteil des Folds.



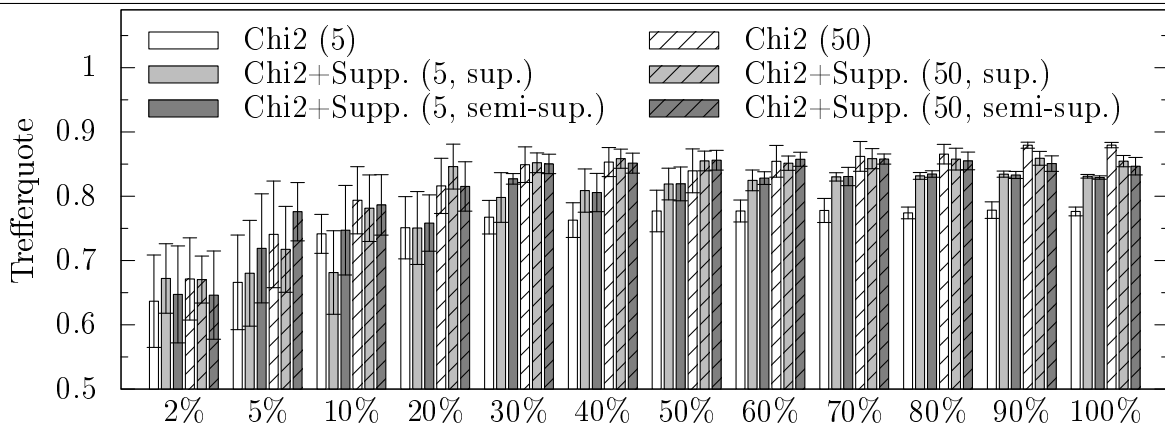
Ergebniskurve 20 Lernkurve für den NCTRER-Datensatz, TSVM, Kreuzvalidierung. 10-fache 10-Fold CV, Prozente bedeuten Anteil des Folds.



Ergebniskurve 21 Lernkurve für den FONTAINE-Datensatz, Chi-2 mit und ohne Supportschränke (0.2). Die Zahl in Klammern ist die Zahl der verbleibenden Merkmale. 10-fache Kreuzvalidierung mit 10 Folds.



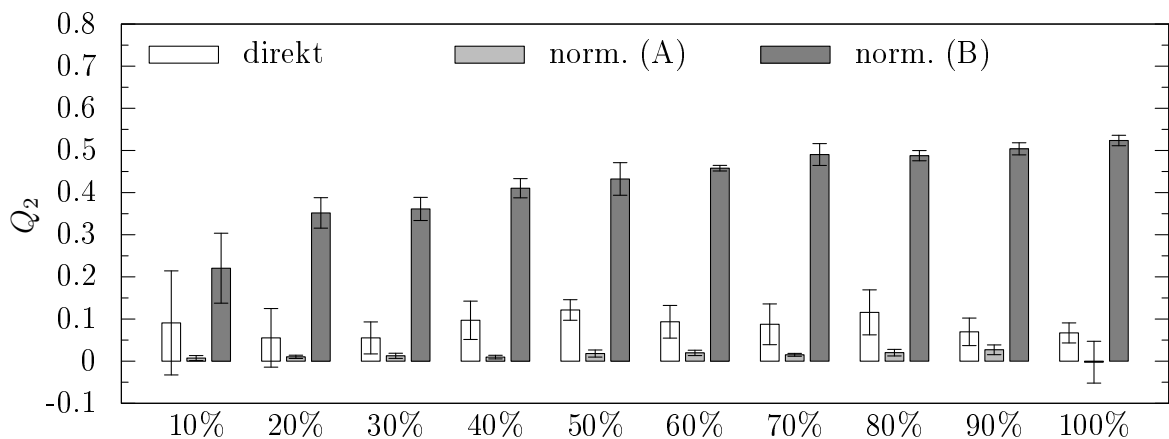
Ergebniskurve 22 Lernkurve für den FONTAINE-Datensatz, Chi-2 mit und ohne Supportschränke (0.3). Die Zahl in Klammern ist die Zahl der verbleibenden Merkmale. 10-fache Kreuzvalidierung mit 10 Folds.



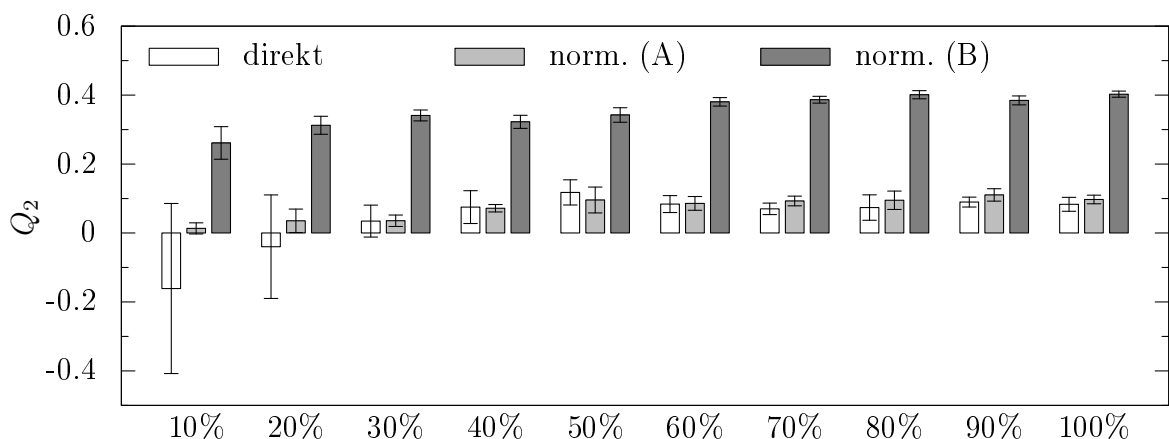
Anhang B

Ergebnisse - Regression

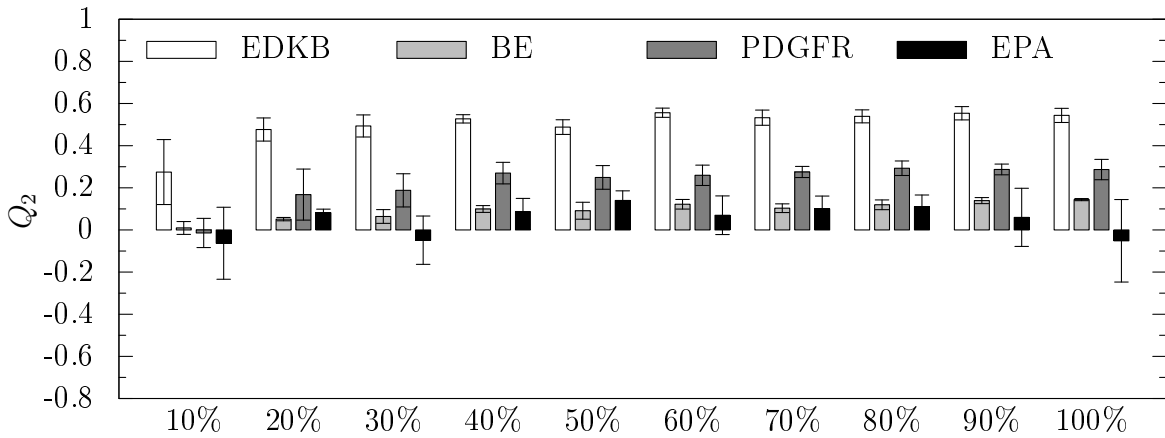
Ergebniskurve 23 Lernkurven für den EPA-Datensatz, direkte Auswertung und Auswertung nach Normierung der Etiketten, wobei zur Auswertung entweder die ursprünglichen (A) oder die transformierten (B) Etiketten verwendet wurden (nu-SVM). 5-fache 5-Fold CV, Prozente bedeuten Anteil des Folds.



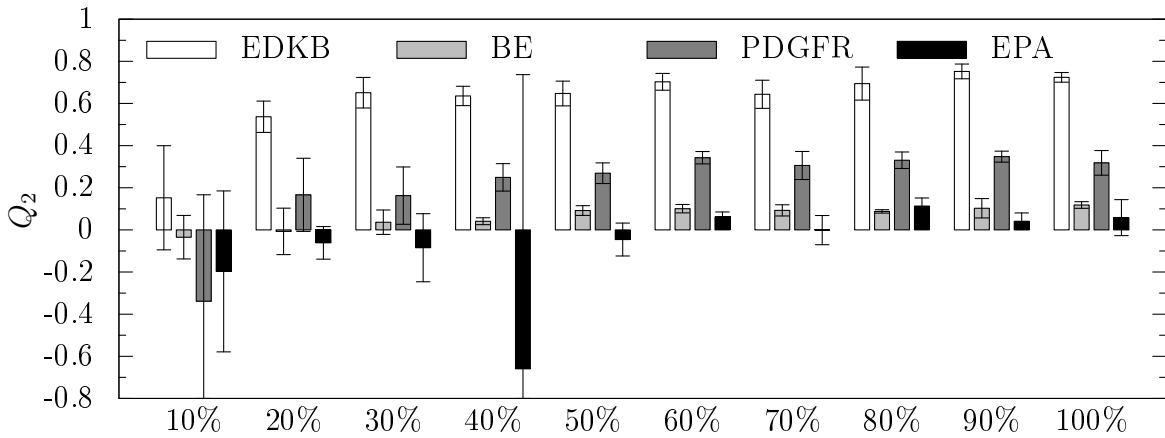
Ergebniskurve 24 Lernkurven für den COX-Datensatz, direkte Auswertung und Auswertung nach Normierung der Etiketten, wobei zur Auswertung entweder die ursprünglichen (A) oder die transformierten (B) Etiketten verwendet wurden (nu-SVM). 5-fache 5-Fold CV, Prozente bedeuten Anteil des Folds.



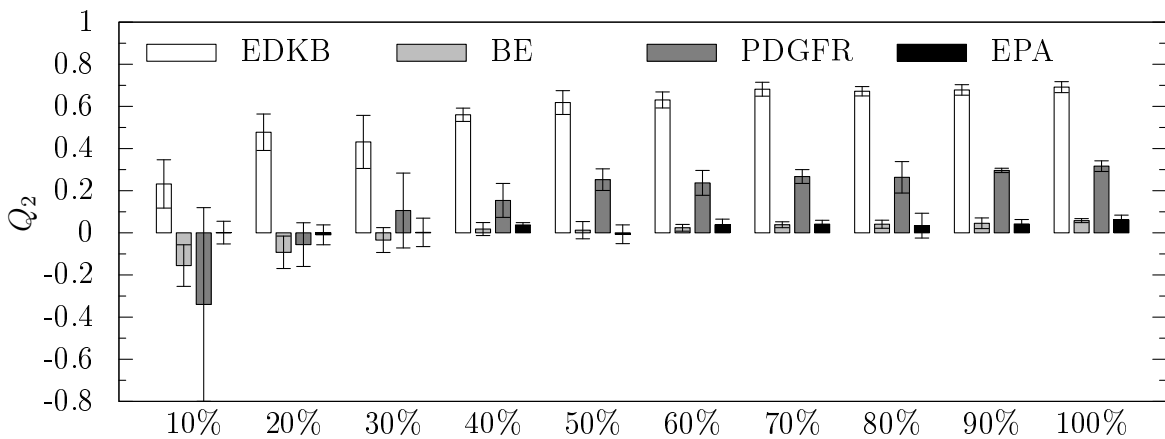
Ergebniskurve 25 Lernkurven für EDKB, BERGSTROM, EPA und PDGFR (kNN, $k = 50$). 5-fache 5-Fold CV, Prozente bedeuten Anteil des Folds.



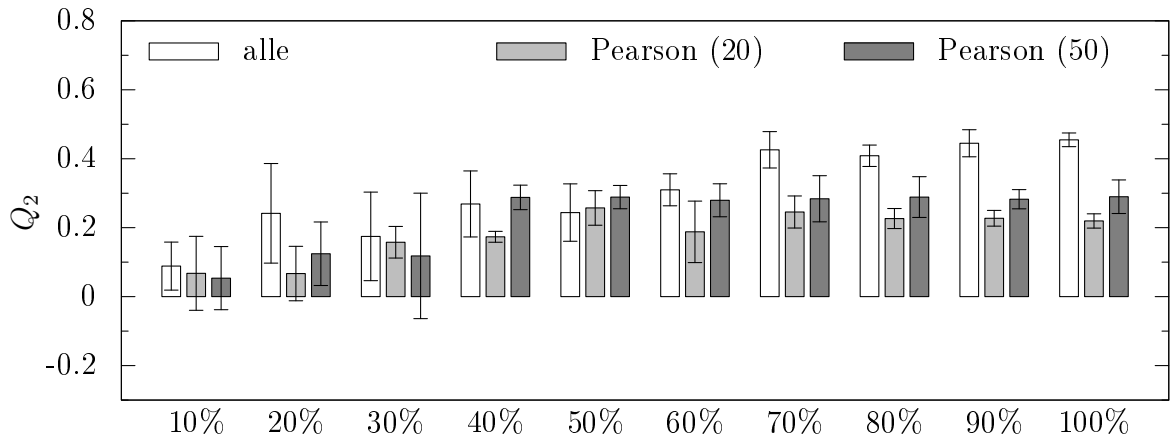
Ergebniskurve 26 Lernkurven für EDKB, BERGSTROM, EPA und PDGFR (direkte Regularisierung). 5-fache 5-Fold CV, Prozente bedeuten Anteil des Folds.



Ergebniskurve 27 Lernkurven für EDKB, BERGSTROM, EPA und PDGFR (Regularisierung mit lin. Kernel). 5-fache 5-Fold CV, Prozente bedeuten Anteil des Folds.



Ergebniskurve 28 Vergleich verschiedener Merkmalsuchverfahren (nu-SVM, PDGFR). Die Zahlen in Klammern entsprechen der Zahl verbleibender Attribute. 5-fache 5-Fold CV, Prozente bedeuten Anteil des Folds.



Ergebniskurve 29 Vergleich verschiedener Parameter für MI-Merkmalsuche (nu-SVM, PDGFR). Hier ist h die Breite des Kerns, k die Anzahl der verbleibenden Merkmale. 5-fache 5-Fold CV, Prozente bedeuten Anteil des Folds.

