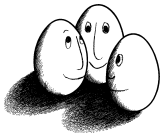
Diplomarbeit

Erweiterung evolutionärer Merkmalskonstruktionen um Case Base Retrieval

Michael Nöthe



Diplomarbeit
am Fachbereich Informatik
der Universität Dortmund

Dortmund, 28. Januar 2008

Betreuer:

Prof. Dr. Katharina Morik
Dipl.-Inform. Ingo Mierswa

Danksagung

Ich bedanke mich bei Prof. Dr. Katharina Morik und Dipl.-Inform. Ingo Mierswa, die mich während meiner Diplomarbeit betreut und durch ihre Anmerkungen, Ratschläge und konstruktive Kritik sehr unterstützt haben. Weiterhin gilt mein Dank Dipl.-Inform. Reza Eslami und Dipl.-Ing. Jörg Monka, die meine Arbeit Korrektur gelesen haben sowie meinen Eltern und Freunden. Schließlich bin ich auch meiner Freundin Sonja zu unendlichem Dank verpflichtet, da sie mich mit großer Geduld und kleinen Denkanstößen immer wieder auf den richtigen Weg gebracht hat.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Die Bestimmung komplexer Zusammenhänge in Datenmengen ist ein Kernthema des maschinellen Lernens. Das Wissen um solche Zusammenhänge kann von vielfältigem Nutzen sein: So kann die Information, unter welchen Umständen und mit welchen Einstellungen eine Maschine am wenigsten Ausschuß produziert, zur Optimierung ihrer Betriebsparameter verwendet werden.

Eine andere Verwendungsmöglichkeit von erkannten Zusammenhängen ist die Prognose, also die Voraussage zukünftiger Entwicklungen aufgrund von Daten der Vergangenheit. Egal, ob der morgige Weltmarktölpreis, die Entwicklung eines Aktienkurses in den nächsten Stunden oder die Kreditwürdigkeit eines Bankkunden prognostiziert werden soll, stets liegt die gleiche mathematische Aufgabe zu Grunde. Aus bereits vorhandenen Daten sollen Zusammenhänge bestimmt werden, mit deren Hilfe sich dann zukünftige Ergebnisse möglichst genau bestimmen lassen. Zur Verdeutlichung werden die gerade genannten Beispiele etwas genauer ausgeführt.

Mögliche Daten, die zur Bildung einer Ölpreisprognose herangezogen werden könnten, sind z.B. die Ölpreise der letzten 200 Tage. Daraus ließen sich Datensätze erzeugen, indem man jeweils die Ölpreise für sieben aufeinanderfolgende Tage in Beziehung setzt zum Preis am achten Tag. Aus den so gewonnenen Datensätzen läßt sich dann ein Zusammenhang bestimmen, anhand dessen möglicherweise der Ölpreis von morgen aus den Tagespreisen der vergangenen Woche prognostiziert werden kann. Für die Voraussage eines Aktienkurses kann etwa aufgrund betriebswirtschaftlicher Kennzahlen zum Unternehmen und der gegenwärtigen Situation am Aktienmarkt eine Prognose vorgenommen werden. Die Bonitätsprüfung wird ebefalls aufgrund bestimmter vergangenheitsbezogener Merkmale wie z.B. dem Einkommen der letzten Jahre und der Kredithistorie durchgeführt. Allen diesen Beispielen ist gemein, dass zunächst ein Modell aus den vorliegenden Daten gebildet wird, dies ist der bereits erwähnte Zusammenhang zwischen bestimmten Ausgangsgrößen und einer Zielgröße. Ist das Modell erst erstellt, kann es im zweiten Schritt zur Optimierung oder Prognose genutzt werden.

Es gibt eine Vielzahl von maschinellen Lernverfahren, die verschiedene Modelle aus Datensätzen erzeugen. Einige Beispiele sind

- Künstliche Neuronale Netze [Hecht-Nielsen 1990]
- Support Vector Machines [Vapnik 1995]
- Entscheidungsbaumlernen [Breiman et al. 1984, Quinlan 1993]
- verschiedene Regressionsverfahren [Draper und Smith 1966]
- Genetische Programmierung [Koza 1996, Banzhaf et al. 1998]

Diese Verfahren haben spezifische Vor- und Nachteile: Die genetische Programmierung etwa leidet an numerischer Instabilität und großem Rechenaufwand. Regressionsanalysen sind zwar relativ effizient, haben aber i.d.R. sehr eingeschränkte und vorher festzulegende Modellklassen (z.B. lineare oder quadratische Funktionen). KNN und SVM sind in dieser Hinsicht zwar flexibler, erzeugen dann aber Modelle, die für einen menschlichen Analysten kaum noch interpretierbar sind. Außerdem ist ihre Ergebnisgüte, wie auch die vieler andere Lernverfahren, stark von der Wahl ihrer verfahrensspezifischen Parameter abhängig.

Eine mögliche Abhilfe für einige dieser Probleme sind Meta-Lernverfahren, wie sie in [Thrun und O'Sullivan 1996, Brazdil et al. 2003] beschrieben werden. Das Problem, das in dieser Arbeit näher betrachtet wird, ist das der eingeschränkten Modellklassen. Als Beispiel soll hier die lineare Regression dienen. Dieses Verfahren stößt bei nichtlinearen Zusammenhängen sehr deutlich an seine Grenzen (siehe Abbildung 1.1 für ein eindimensionales Beispiel).

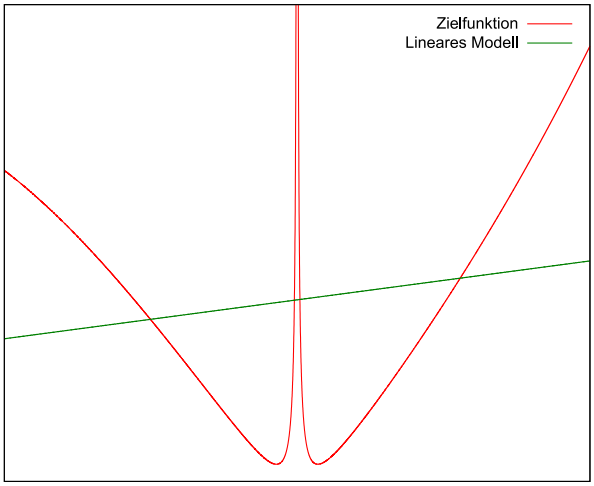


Abbildung 1.1.: Zielfunktion und Modell eines linearen Lerners

Ein Verfahren, mit dem sich die Modellklasse vergrößern läßt, ist die Merkmalserzeugung (Feature Generation). Durch Hinzufügung von neuen Attributen zum vorhandenen Datensatz wird dem eigentlich linearen Lernverfahren die Möglichkeit gegeben, einen höherdimensionalen und nicht mehr nur linearen Modellraum zu durchsuchen. Abbildung 1.2 zeigt, wie die Hinzugabe eines quadratischen Attributes das gelernte Modell verbessert im Vergleich zur vorherigen Abbildung 1.1.

Der in dieser Arbeit verwendete Feature Generator basiert auf einem evolutionären Algorithmus, ähnlich dem der genetischen Programmierung. Ein Nachteil dieser Klasse von Algorithmen ist der hohe Laufzeitaufwand, der durch die Bearbeitung der Populationen über viele Generationen hinweg anfällt. Daher wird ein dem fallbasierten Schließen ähnlicher Ansatz entwickelt und getestet, der die Konvergenz des evolutionären Feature Generators beschleunigen soll. Für eine große Klasse von Regressionsaufgaben sollen vorgefertigte Merkmalskonstrukte in einer Fallbasis abgelegt werden. Dadurch soll erreicht

werden, dass das lineare Lernverfahren eine neue Regressionsaufgabe mit Hilfe dieser vorgefertigten Merkmale schnell und gut löst, ohne einen langen Feature Generator-Lauf durchführen zu müssen. Dabei wird besonderes Augenmerk auf die kompakte Darstellung und effiziente Abfrage der Case Base gerichtet.

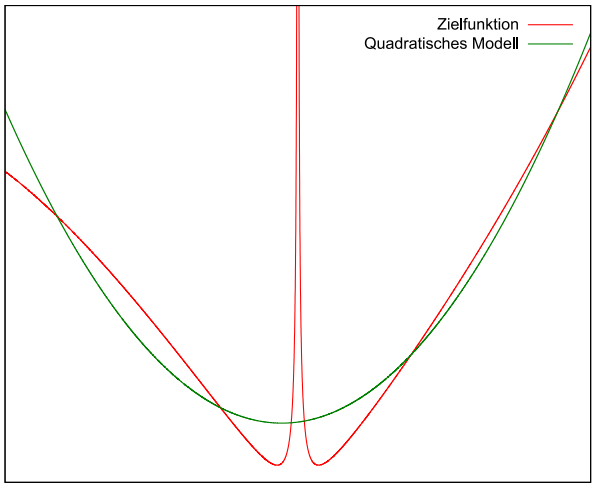


Abbildung 1.2.: Zielfunktion und Modell mit quadratischem Feature

Ziel dieser Arbeit ist es, den sowohl in [Mierswa und Wurst 2005b] als auch in [Mierswa und Wurst 2005a] vorgeschlagenen Ansatz zur Case Base-unterstützten Feature Construction theoretisch und praktisch zu evaluieren, verschiedene Varianten der Anwendung zu erproben und schließlich eine Möglichkeit zur breiteren Anwendbarkeit zu untersuchen.

Zunächst werden jedoch in Kapitel 2 die in dieser Arbeit benutzten Techniken und Begriffe eingeführt. Dort wird geklärt, was im Rahmen dieser Arbeit unter Lernaufgaben und -verfahren verstanden wird. Weiterhin wird auf die evolutionäre Merkmalserzeugung und die Grundlagen des Case Base-Ansatzes eingegangen. Kapitel 3 befasst sich mit den theoretischen Hintergründen der Basisattributgewichtung, während in Kapitel 4 auf Basis dieser Gewichtung die Ähnlichkeit zwischen zwei Lernaufgaben behandelt wird. Dabei wird neben den in [Mierswa und Wurst 2005a] behandelten Ähnlichkeitsmaßen eine Reihe weiterer Maße auf ihre Eignung für den verfolgten Ansatz hin überprüft. In Kapitel 5 wird ein neues Abstandsmaß auf Basis von konstruierten Attributen eingeführt, das sich insbesondere für die Verwendung bei evolutionären Feature Generatoren eignet. Dabei wird von bereits generierten Merkmalen auf die Ähnlichkeit zwischen zwei Problemfällen geschlossen. Um die experimentelle Erprobung der bis dahin vorgestellten Techniken wird es in Kapitel 6 gehen. Darin wird eine Vielzahl von Versuchen beschrieben, mit denen zwei verschiedene Anwendungsarten des Case Base-Ansatzes sowie der Einfluss mehrerer Parameter evaluiert werden. In Kapitel 7 wird schließlich eine Erweiterung des Ansatzes entwickelt und getestet, die seine breitere Anwendung ermöglichen kann. Während zuvor alle Lernprobleme in der Case Base die gleichen Attribute haben mussten, ist es nun möglich, auch Problemfälle mit gänzlich verschiedener Attributsignatur als Lösungshilfe

heranzuziehen. Abgeschlossen wird diese Arbeit von einer Zusammenfassung, in der der Case Base-Ansatz bewertet wird und seine Vor- und Nachteile gegenübergestellt werden. Weiterhin wird ein Ausblick gegeben auf Komponenten des Ansatzes, deren Weiterentwicklung in der Zukunft als lohnend erscheint.

2. Allgemeine Grundlagen

In diesem Kapitel soll zunächst der Bereich der Lernverfahren eingeführt werden. Darauf folgt unter Erläuterung der technischen Voraussetzungen die Einbettung des fallbasierten Schließens in diesen Bereich.

2.1. Lernaufgaben

Wie schon in der Einleitung erläutert, ist die grundlegende Aufgabe, die in dieser Arbeit verfolgt wird, die Bestimmung von Zusammenhängen zwischen verschiedenen Größen in einem Datensatz. Dabei ist - zumindest für die im Weiteren zu besprechenden Lernverfahren - bereits vorgegeben, welches die Ausgangsgrößen sind und welche die Zielgröße ist. Zunächst muss also definiert werden, was im Weiteren unter einem Datensatz verstanden wird:

Definition 1 *Ein **Datensatz** t besteht aus einer Menge von Datenpunkten oder Instanzen, die wiederum die Form $(x, y) \in (X \times Y)$ haben. X bezeichnet den Merkmalsraum, x ist der Vektor der Basisattribute oder Merkmale und Y die Menge der möglichen Zielwerte bzw. Label.*

Für die Zwecke dieser Arbeit werden die Begriffe Datensatz, Trainingsdaten, Lernproblem und Lernaufgabe synonym verwendet.

Der in Tabelle 2.1 angedeutete Datensatz greift das Beispiel der Ölpreisprognose aus der Einleitung wieder auf. In der Tabelle sind die Ölpreise 7, 3 und einen Tag vor dem Stichtag als Basisattribute und der Preis am Stichtag als Zielgröße eingetragen.

Preis 7 Tage vorher	3 Tage vorher	1 Tag vorher	Stichtag
78 $	83 $	85 $	86 $
69 $	64 $	82 $	91 $
83 $	81 $	86 $	86 $
58 $	71 $	72 $	71 $
⋮	⋮	⋮	⋮

Tabelle 2.1.: Beispieldatensatz mit Ölpreisen jeweils zu einem Stichtag und den vorhergegangenen Tagen

2.2. Lernverfahren

Das Ziel von überwachten Lernverfahren im Allgemeinen ist es, aus einer Menge von Trainingsdaten $t \subset (X \times Y)$ mit einer unbekannten funktionalen Beziehung zwischen $x \in X$ und $y \in Y$ eine Abbildungsvorschrift $f : X \rightarrow Y$ zu generieren, die die unbekannte Beziehung so gut wie möglich annähert. Diese Abbildungsvorschrift kann dann zur Vorhersage von unbekannten Beispielen benutzt werden. Ein klassisches Beispiel für solche Verfahren ist die lineare Regression, die hier kurz vorgestellt werden soll, da sie im weiteren Verlauf dieser Arbeit noch benutzt wird.

2.2.1. Lineare Regression

Wie der Name bereits andeutet, setzt dieses Verfahren einen linearen funktionalen Zusammenhang der Form

$$y = bx + k$$

mit $y \in \mathbb{R}, x \in \mathbb{R}^n$ auf den gegebenen Trainingsdaten voraus. Falls in den Trainingsdaten tatsächlich ein solcher Zusammenhang besteht, könnte man auf triviale Weise mittels $n + 1$ Datenpunkten die korrekte Geradengleichung bestimmen. Dies ist jedoch in der Praxis selten der Fall. Zum einen können Meßfehler bei der Ermittlung der Zielwerte aufgetreten sein, zum anderen besteht auch die Möglichkeit, dass kein linearer Zusammenhang zwischen x und y existiert. Diesem Problem kann z.B. durch die Benutzung der Ordinary Least Squares Methode (Prinzip der kleinsten Fehlerquadrate) bei der Bestimmung der Regressionsgerade begegnet werden. Die Anwendung dieser Methode wird u.a. in [Kreyszig 1975, Sachs 1997] für Datensätze mit nur einer unabhängigen Variable beschrieben. Dabei wird versucht, eine Regressionsgerade $y = bx + k$ zu finden, für die die Gesamtabweichung aller Datenpunkte (x_i, y_i), $i = 1..m$ von dieser Geraden minimiert wird. Dabei ist zunächst der vertikale Abstand der Datenpunkte von der Regressionsgeraden zu bestimmen. Dieser berechnet sich als

$$|y_j - bx_j - k|,$$

und daraus ergibt sich die zu minimierende Summe der Fehlerquadrate

$$a = \sum_{i=1}^{m} (y_i - bx_i - k)^2.$$

Für die Minimierung müssen die beiden Ableitungen der Formel nach b und k gleich null gesetzt werden. Nach Auflösung des so erhaltenen Gleichungssystems ergeben sich

$$b = \frac{\sum_{i=1}^{m} x_i y_i \; - n\overline{x}\,\overline{y}}{\sum_{i=1}^{m} x_i^2 \; - n\overline{x}^2}$$

und

$$k = \overline{y} - b\overline{x},$$

wobei $\overline{x}$ bzw. $\overline{y}$ jeweils die Mittelwerte über die x- bzw. y-Werte darstellen. Für die ausführliche Herleitung sei auf [Kreyszig 1975], S. 266ff. verwiesen. Obwohl hier nur

für eine unabhängige Variable erläutert, läßt sich diese Methode auch auf Datensätze mit mehreren Variablen erweitern.

Der Vorteil der linearen Regression liegt vor allem in ihrer effizienten Berechenbarkeit. Dies ist für diese Arbeit besonders wichtig, da sie im weiteren Verlauf in ein Meta-Lernverfahren eingebettet und dort wiederholt angewendet wird. Weitere lineare Lernverfahren sind die Pace-Regression des Weka-Projektes [Wang und Witten 1999], die ihren Entwicklern zufolge besonders gut mit irrelevanten Variablen umgehen kann, sowie die lineare Support Vector Machine von [Vapnik 1995], die besonderen Wert auf die Generalisierungsfähigkeit legt.

Allen diesen bislang vorgestellten Lernverfahren ist jedoch gemein, dass sie auf lineare Modelle festgelegt sind. Ein weiteres Problem, dass auch Verfahren wie künstliche neuronale Netze, Support Vector Machines mit komplexeren Kernfunktionen oder Genetische Programmmierung betrifft, ist etwa die Wahl der Parameter.

2.2.2. Meta-Lernverfahren

Um diese Lernverfahren zu verbessern, kann es sinnvoll sein, verschiedene Arten von Vorverarbeitungsschritten durchzuführen. Eine Art von Vorverarbeitung ist die Bestimmung von optimalen Werten für die verfahrensspezifischen Parameter, wie z.B. Kerneltyp, C und ϵ für Support Vector Machines. Eine andere Methode, um ein Lernverfahren zu unterstützen, ist die explizite Anpassung des Eingaberaumes X. Dadurch ist es möglich, u.a. das Problem der beschränkten Modellklassen einiger Lernverfahren zu beheben.

Da die optimalen Parameterwerte bzw. der am besten geeignete Eingaberaum in der Regel nicht a priori bekannt sind und nicht ohne weiteres berechnet werden können, benutzt man ein Meta-Lernverfahren, um sie zu bestimmen. Das Meta-Lernverfahren besteht in diesem Fall aus der wiederholten Anwendung des eigentlichen Lernverfahrens (im Folgenden "inneres Lernverfahren" genannt). Anders ausgedrückt wird das innere Lernverfahren mit verschiedenen Parametersätzen bzw. verschiedenen Eingaberäumen mehrmals gestartet und die Ergebnisse der Läufe evaluiert, um die Meta-Suche zu steuern.

2.3. Feature Selection und Feature Generation

Während die Parameteroptimierung z.B. mittels neuronaler Netze in [Cicirello 2000] oder Genetischer Algorithmen in [Abraham 2003] abgehandelt wird, wird es im Folgenden um den zweiten Fall der Meta-Optimierung, die Anpassung des Eingaberaumes, gehen. Die dazu verwendeten Verfahren sind Feature Selection und Feature Generation.

Die einzelnen Dimensionen dieses Raumes werden auch als Features oder Attribute bezeichnet, die Dimensionen des originalen Eingaberaumes als Base Features oder Basisattribute. Warum kann es sinnvoll sein, diesen Raum zu verändern? Wir betrachten zunächst eine Möglichkeit zur Dimensionsreduktion des Eingaberaumes, die Feature Selection ([Yang und Honavar 1997, Liu et al. 2002, Guyon und Elisseeff 2003]). Es kann sich herausstellen, dass nicht alle Features den gleichen Einfluss auf die Ergebnisvariable Y haben. Features, die auf den Trainingsdaten nur eine sehr geringe oder gar keine Korrelation mit der Ergebnisvariable aufweisen, können als irrelevant für die Lernaufgabe angesehen werden. Ihre Entfernung aus dem Eingaberaum sorgt zum einen für eine

Reduktion der Berechnungskomplexität des inneren Lernverfahrens, zum anderen dient sie der Rauschreduktion. Die als irrelevant identifizierten Features könnten, falls sie nicht entfernt werden, das innere Lernverfahren dazu veranlassen, eine unnötig komplizierte und schlechter generalisierende Vorhersagefunktion zu lernen.

Dem entgegen steht die künstliche Erzeugung von Features. Wenn das innere Lernverfahren z.B. nur in der Lage ist, eine lineare Funktion zu berechnen, die Trainingsdaten aber nichtlineare Zusammenhänge enthalten, so kann es stark zur Konvergenz des inneren Lernverfahrens beitragen, wenn der Eingaberaum um passende nichtlineare Features erweitert wird. Diese neuen Features werden durch z.B. durch die multiplikative Verknüpfung von Base Features oder die Anwendung nichtlinearer Funktionen (z.B. Sinus, Cosinus, Wurzel, etc.) erzeugt. Dieser Ansatz wird als Feature Generation bezeichnet [Ritthoff et al. 2002]. Es sollte an dieser Stelle angemerkt werden, dass auch die Verwendung von kernelbasierten Verfahren ([Mierswa 2006, Rodríguez 2004, kopf et al. 1999]) den Eingaberaum erweitert und in diesem Sinne als implizite Feature Generation betrachtet werden kann. Der Unterschied zur hier gemeinten expliziten Feature Generation ist jedoch der, dass Kernelmethoden die Dimension des Eingaberaumes um ganze Funktionsräume erweitert. Bei der Verwendung des polynomiellen Kernels $k(x, x') = (\langle x, x' \rangle + 1)^p$ beispielsweise besteht die Menge der impliziten Features aus allen Monomen des verwendeten Polynoms ([Drucker et al. 1997]). Wenn d die Dimension des Eingaberaumes ist und p der Grad des Polynoms, dann ergeben sich daraus immerhin $\binom{d+p}{p}$ Features. Der Nachteil dieser impliziten Feature Construction ist jedoch der, dass diesen Features keinerlei explizite Bedeutung mehr zugeordnet werden kann.

Um die Bedeutung der für eine Lernaufgabe explizit erzeugten bzw. ausgewählten Features zu veranschaulichen, erscheint es sinnvoll, den Zusammenhang zwischen innerem Lernverfahren und der Menge der Features zu betrachten. Die Aufgabe der hier betrachteten Lernverfahren ist wie schon erwähnt die Bestimmung einer Abbildungsvorschrift $f : X \to Y$, die auch als Modell bezeichnet wird. Diese Abbildungsvorschrift kann o.B.d.A. als Funktionsbaum gegeben sein. Die Erzeugung dieses Funktionsbaumes teilt sich bei den oben eingeführten Meta-Lernverfahren in zwei Bereiche: Zum einen wird durch das innere Lernverfahren bestimmt, wie die Wurzel bzw. der obere Bereich des Baums aufgebaut ist. Hier werden die darunter hängenden Teilbäume durch die vom inneren Lernverfahren erzeugte Funktion verknüpft. Zum anderen stellen die am oberen Bereich hängenden Teilbäume die von der Gesamtfunktion verwendeten Features dar. Im Falle von konstruierten Features kann es sich dabei tatsächlich um komplexere Teiläume handeln, während Base Features durch einzelne Blattknoten dargestellt werden. Abbildung 2.1 zeigt ein Beispiel für ein Modell eines linearen inneren Lernverfahrens. Der grau unterlegte Bereich umfaßt den oberen Bereich des Baumes, den das innere Lernverfahren bestimmt. Es addiert die gewichtete Summe der als Teilbäume T_1 bis T_m dargestellten Features. Bei T_1 bis T_m kann es sich um einfache Basisattribute, aber auch um zusammengesetzte Features mit theoretisch beliebiger Komplexität handeln. Beispiele für Lernverfahren, die eine solche lineare "Funktionswurzel" lernen, sind die Lineare Regression und die Support Vector Machine (siehe [Vapnik 1995]) mit Skalarprodukt-Kernel. Denkbar sind an dieser Stelle auch nichtlineare Verfahren wie etwa SVMs mit polynomiellem, RBF- oder anderen Kerneln. Viele Lernaufgaben sind allein mit linearen Funktionen nicht gut genug zu lösen, so dass man sich in der Regel bei einem Meta-Lernverfahren

entscheiden muss, in welchem Bereich des Funktionsbaumes man mit Nichtlinearität arbeiten will. Entweder benutzt man ein nichtlineares inneres Lernverfahren, oder man konstruiert nichtlineare Features. Die Einbeziehung von Nichtlinearität in beide Bereiche scheint zumindest offensichtlich keinen Sinn zu ergeben, da sich dadurch die Klasse der erlernbaren Funktionen nicht vergrößert. Da sich der in dieser Arbeit verfolgte Ansatz von [Mierswa und Wurst 2005b] vor allem mit der Erzeugung und Wiederverwendung von für eine Lernaufgabe besonders geeigneter und durchaus komplexer Features widmet, wird dort als inneres Lernverfahren eine Lineare Regression benutzt, um die einzelnen Features zu einer Gesamtfunktion zu verknüpfen. Durch die Benutzung expliziter Attribute in Form von geschlossenen Funktionalen ist gewährleistet, dass die im Ablauf des Lernverfahrens erzeugte Gesamtfunktion ebenfalls als geschlossenes Funktional vorliegt. Das Modell, dass schließlich erzeugt wird, läßt sich graphisch als Funktionsbaum oder textuell als Term notieren und steht somit dem Benutzer in expliziter und nachvollziehbarer Form zur Verfügung. Ein solcher Funktionsbaum ist in Kapitel 6 Abbildung 6.1 dargestellt.

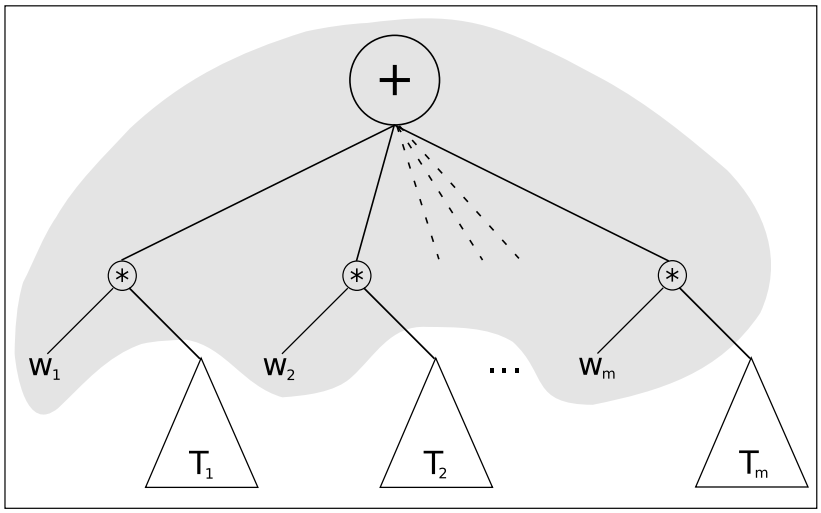


Abbildung 2.1.: Funktionsbaum mit linearer Wurzel

2.4. Evolutionäre Feature Generation

Eine Methode zur Suche nach einem geeigneten Eingaberaum für eine Lernaufgabe besteht in der Benutzung eines evolutionären Feature Generators [Ritthoff et al. 2002]. Ein solcher Algorithmus stellt eine Weiterentwicklung des klassischen genetischen Algorithmus zur Feature Selection [Yang und Honavar 1997] dar: Innerhalb einer Schleife wird ein Menge (Population) von Attributräumen (Individuen) evolviert. Dabei werden mehrere Evolutionsoperatoren verwendet, die dazu dienen, die Individuen zu verändern und so die Suche nach einem optimalen Eingaberaum voranzutreiben. Der Mutationsoperator wird auf ein Individuum angewendet und erzeugt durch zufallsgesteuertes Hinzufügen oder Entfernen von Attributen ein neues, leicht verändertes Individuum. Der

Crossoveroperator kreuzt zwei Individuen. Dabei wird jeweils ein Teil der Attribute dieser beiden Individuen auf ein neues Individuum übertragen. Der Feature Generator-Operator arbeitet im Gegensatz dazu auf der Ebene einzelner Attribute: Ein oder mehrere Attribute eines ausgewählten Individuums werden durch einen atomaren Operator wie etwa Summe, Differenz, Produkt, Quotient, Exponentiation, etc. zu einem neuen Attribut verknüpft, welches dem Ausgangsindividuum hinzugefügt wird. Schlussendlich werden alle Attributräume des aktuellen Schleifendurchlaufs (Generation) bewertet und einer auf dieser Bewertung aufbauenden Selektion unterworfen. Die selektierten Individuen werden dann in die nächste Generation übernommen und weiter evolviert. Die Bewertung der Attributräume findet dabei durch die Anwendung eines inneren (linearen) Lerners statt. Die Performanz, die der innere Lerner mit Hilfe des jeweiligen Attributraumes erzielt, wird als Fitnesswert des zugehörigen Individuums zur Selektion benutzt. Die Schleife der evolutionären Suche ist mit einem Abbruchkriterium versehen, in der Regel eine feste Zahl an Generationen oder eine Zielperformanz, die vom besten Individuum erreicht werden muss.

2.5. Case Base-Ansatz

Die klassische (explizite) Feature Generation verbraucht als umschließendes Lernverfahren viel Rechenzeit, da sie auf eine Vielzahl von Durchläufen des inneren Lernverfahrens angewiesen ist. Um die Suche nach geeigneten Features abzukürzen, wird in [Mierswa und Wurst 2005b] die Wiederverwendung von bereits konstruierten Features vorgeschlagen. Die dort erarbeitete Vorgehensweise orientiert sich am Ansatz des fallbasierten Schließens (Case Based Reasoning). Der allgemeine CBR-Ansatz sieht vor, die Lösung für bereits erfolgreich bearbeitete Probleme in einer Fallbasis abzuspeichern und sie unverändert oder adaptiert wiederzuverwenden, wenn man erneut vor dem gleichen oder einem ähnlichen Problem steht. Ein Datensatz, der aus dem Problem und seiner Lösung besteht, wird Fall oder Case genannt. Der hier verfolgte spezielle Case Base-Ansatz sieht nun vor, die klassische Feature Generation durch den Abruf passender, bereits konstruierter Features aus der Case Base zu unterstützen. Diese Features stellen dabei Lösungen von bereits mit einem Feature Generator bearbeiteten Problemen dar. Abbildung 2.2 stellt den Ablauf am Beispiel eines evolutionären Feature Generators graphisch vor. Den üblichen evolutionären Operatoren wie Mutation und Crossover wird ein weiterer hinzugefügt, der eine Anfrage an die Case Base stellt und die Antwort in Form von konstruierten Features in den evolutionären Algorithmus mit einfließen läßt. Wie dieser Ablauf im Detail aussieht, wird in den nächsten Kapiteln beschrieben.

Jede Implementierung des CBR-Ansatzes verlangt eine konkrete Spezifizierung folgender Punkte (siehe Abbildung 2.3):

- Kodierung des Problems und der zugehörigen Lösung
- Suche nach Fällen, die einem neuen Problem ähnlich sind
- Anpassung der vorgeschlagenen Lösung(en) auf das neue Problem
- Entscheidung, ob ein neues Problem so gut gelöst wurde, dass es in die Fallbasis aufgenommen werden kann

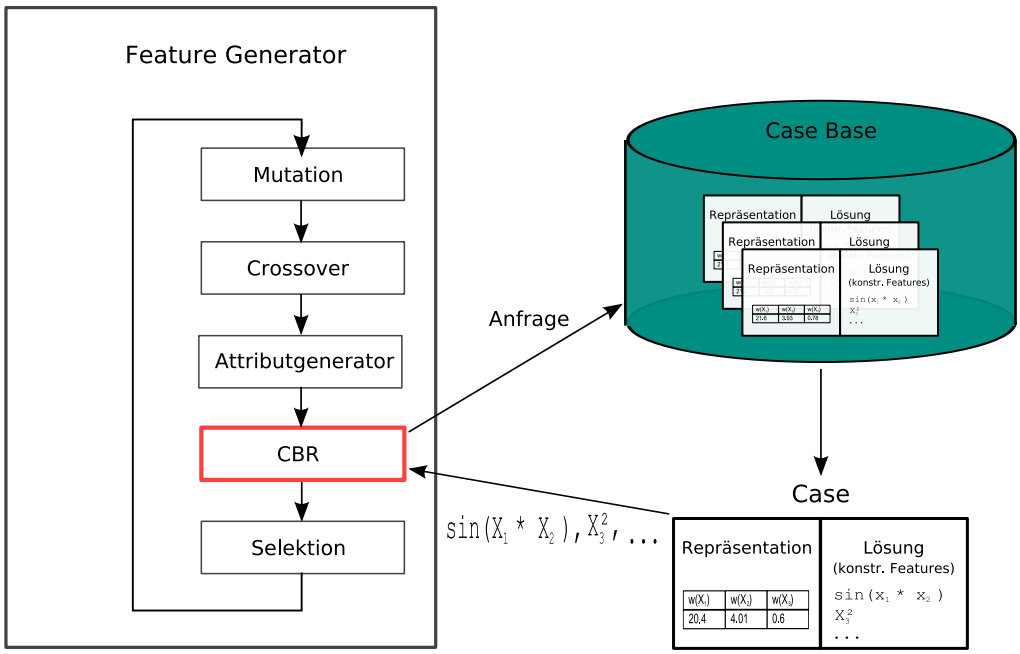


Abbildung 2.2.: Feature Construction mit CBR-Unterstützung

Insbesondere auf die ersten beiden Punkte wird in den folgenden Kapiteln eingegangen, wohingegen die Adaption der angebotenen Lösungen zunächst lediglich darin besteht, die konstruierten Features mit in die Evaluierung des Feature Generators aufzunehmen. Die Bewertung einer gefundenen Lösung ergibt sich dann in der Regel aus dem Meta-Lernverfahren, welches der Case Base-Ansatz unterstützt. Konvergiert es in seinem Verlauf, so kann man davon ausgehen, eine gute Lösung zu erhalten.

Die Frage der Kodierung eines Falles könnte man trivial so lösen, dass man den Trainingsdatensatz als Problembeschreibung benutzt. Dies hätte jedoch zwei gravierende Nachteile: Zunächst einmal wird die Beschreibung auf diese Art viel Speicherplatz erfordern bzw. (in einem verteilten Szenario) die Übertragung einer Anfrage an die Case Base viel Bandbreite verbrauchen, wenn die Trainingsdatensätze auch nur leidlich groß sind. Außerdem wäre es bei dieser Kodierung fraglich, ob ein brauchbares (und effizientes) Abstandsmaß zwischen den Problemen gefunden werden kann. Daher wird in [Mierswa und Wurst 2005b] ein anderer Kodierungsansatz benutzt, der sich auf ein Relevanzmaß für die Basisattribute stützt.

2.6. Grundannahme zum Case Base-Ansatz

In dem oben erwähnten Relevanzmaß kommt die zentrale Annahme, auf die der Case Base-Ansatz aufbaut, zum Vorschein: Wenn eine Menge von Basisfeatures für eine Lernaufgabe wichtig sind, dann sind auch die aus ihren Elementen konstruierten Features

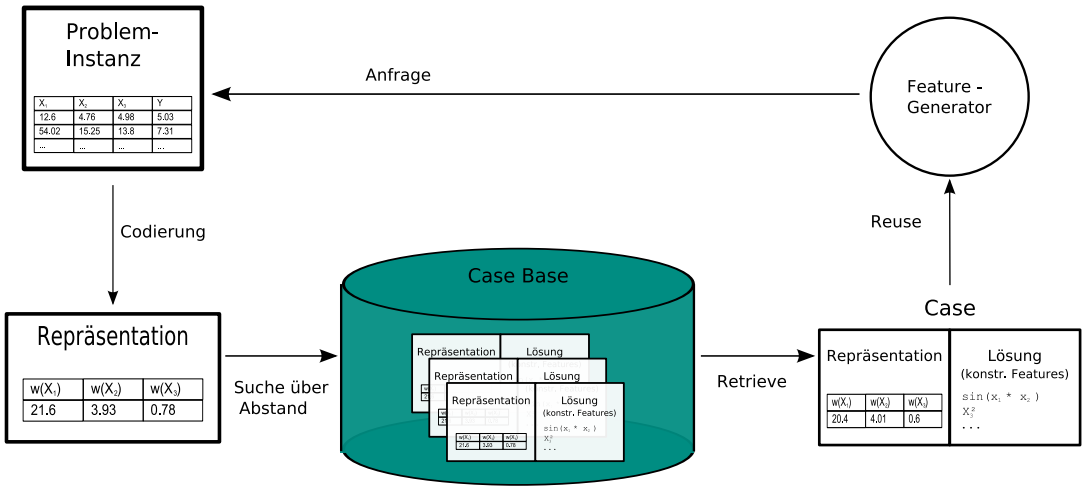


Abbildung 2.3.: Ablauf einer Case Base Anfrage

wichtig und umgekehrt. Ein Beispiel: Wenn das konstruierte Feature

$$\frac{X_{ik}}{sin(X_{il} * X_{im})}$$

sich als relevant für die Lernaufgabe t_i erweist, so sind laut der Annahme auch die Basisfeatures X_{ik}, X_{il} und X_{im} wichtig für t_i.

Die zweite wichtige These, die im Grunde eine etwas genauere Quantifizierung der ersten Annahme darstellt, besagt, dass sich die funktionale Verknüpfung, in der sich ein Basisattribut befindet, in der Wichtigkeit dieses Attributes widerspiegeln sollte. Ein einfaches Beispiel für diese These wäre die Aussage, dass ein Attribut X_{im}, welches in die zu lernenden Funktion $f : X \rightarrow Y$ als Exponent eingeht, ein höheres Gewicht erhalten wird, als ein Attribut X_{in}, welches nur als linearer Summand in f einfließt. f sähe also etwa folgendermaßen aus:

$$f(X) = \cdots + \alpha * X_{in} + \cdots + \beta^{X_{im}} + \cdots$$

Dieses Beispiel läßt bereits erahnen, wie schwierig eine formale Überprüfung dieser Annahmen sein dürfte. Wenn z.B. $\alpha \gg \beta$ gilt oder die Wertebereiche der einzelnen Ausprägungen der Attribute X_{in} und X_{im} stark voneinander abweichen, kann die berechnete Gewichtung der beiden Attribute doch stark von einer Gewichtung abweichen, die man intuitiv aufgrund der Position der Attribute in der Formel für $f(X)$ erwarten würde. Daher kann in dieser Arbeit auch nur durch empirische Untersuchungen (Kapitel 6 und Abschnitt 7.3 überprüft werden, in wie weit die beiden oben vorgestellten Grundannahmen zutreffend sind.

Nichtsdestoweniger stellt diese Annahme den Grundgedanken des gesamten Case Base-Ansatzes dar. Durch sie wird der Schluss von der Repräsentation des aktuellen Falles auf

die Lösung eines ähnlichen Falles erst möglich. Es ist also nötig, dass die hier formulierte Beziehung zwischen den Basisattributen eines Falles und seinen konstruierten Features tatsächlich existiert.

3. Grundlagen der SVM-Featuregewichtung

Thema dieses Kapitels ist die Kodierung von Lernproblemen, die durch einen Trainingsdatensatz gegeben sind, in eine kompaktere Repräsentation zur Abspeicherung in der Case Base. Die Motivation hierfür liegt zum einen in der Speicherplatzersparnis und höheren Geschwindigkeit bei der Verwaltung der Case Base, zum anderen aber auch in der Möglichkeit, die Ähnlichkeitsbestimmung zwischen abgelegten Lernproblemen effizient durchführen zu können (siehe dazu Kapitel 4). Wie schon in Abschnitt 2.5 angedeutet, besteht diese Kodierung in der Relevanzbestimmung der Basisattribute.

3.1. Relevanzbestimmung der Base Features

Um die Bedingungen für das Relevanzmaß einzuführen, sind zunächst einige einfache Definitionen nötig. T sei die Menge aller betrachteten Lernaufgaben. Zu einem einzelnen Lernproblem $t_i \in T$, welches sich aus einem Trainingsdatensatz ergibt, sei X_i der Eingabevektor und Y_i die Ergebnisvariable. Die einzelnen Komponenten des Eingabevektors X_i werden Attribute oder Features genannt. Sie lassen sich aufteilen in eine Menge von Base Features X_B, die für alle betrachteten Lernaufgaben gleich sind und die gleiche Semantik besitzen, und in eine Menge von konstruierten Features $X_i \backslash X_B$. Für diese Features werden sich im Weiteren die folgenden beiden Eigenschaften als wichtig erweisen:

Definition 2 *Ein Feature X_{ik} ist für eine Lernaufgabe t_i* ***irrelevant*** *genau dann, wenn $Pr(Y|X_{ik}) = Pr(Y)$, d.h. wenn es unabhängig von der Ergebnisvariable ist.*

Die Benutzung eines irrelevanten Features X_{ik} führt also für die Lernaufgabe t_i zu keiner Verbesserung. Im Weiteren wird die Menge der für eine Lernaufgabe t_i irrelevanten Features als IF_i bezeichnet.

Definition 3 *Zwei Features X_{ik} und X_{il} heißen* ***alternativ*** *(Notation: $X_{ik} \sim X_{il}$) genau dann, wenn $X_{il} = a + b * X_{ik}$ mit $b > 0$, d.h. X_{ik} ist linear abhängig von X_{il}.*

Zwei für die Lernaufgabe t_i alternative Features sind gegeneinander austauschbar, ohne dass die Güte darunter leidet.

Der Case Base-Ansatz sieht nun folgenden Ablauf vor: Zunächst werden für eine Lernaufgabe $t_i \in T$ die Relevanzen bzw. Gewichte aller Basisfeatures X_B bestimmt. Der so berechnete Gewichtsvektor $W_i(X_B)$ stellt zunächst das einzige Identifikationsmerkmal für die Lernaufgabe t_i dar, anhand dessen eine Aufgabe in der Case Base abgelegt wird. Um nun für eine gegebene Lernaufgabe t_i eine Menge T_{sim_i} von möglichst ähnlichen Aufgaben in der Case Base zu finden, wird ein Abstandsmaß $d(t_i, t_j)$ benötigt, das auf den Gewichtsvektoren definiert ist. Die Aufgaben T_{sim_i} sind in der Case Base mit den für sie

konstruierten Features abgelegt. Die Vereinigung dieser konstruierten Features werden auf das aktuelle Problem t_i angewendet, zusätzlich zu den Features, die ein herkömmlicher Feature Generator erzeugt. Wenn die Aufgabe t_i dann gut gelöst wurde, sollte sie ebenfalls mit den für sie konstruierten Features in der Case Base als neuer Fall abgelegt werden. Dabei ist zu beachten, dass eine neue Aufgabe gerade dann ein interessanter Kandidat zur Aufnahme in die Case Base sein könnte, wenn die konstruierten Features aus der Case Base-Anfrage *nicht* oder *nur wenig* zur Lösung beigetragen haben. Dann nämlich ist davon auszugehen, dass die Case Base um neues Wissen erweitert wird. Denn offensichtlich waren die Probleme, die bisher in der Case Base waren, der aktuellen Lernaufgabe t_i nicht ähnlich genug, um Wesentliches zu ihrer Lösung beitragen zu können.

Im Folgenden werden zunächst die Bedingungen an die Gewichtungsmethode festgelegt, bevor in Kapitel 4 die Anforderungen an das Abstandsmaß formuliert werden:

Bedingungen für die Gewichtungsfunktion 1 *Sei w eine **Gewichtungsfunktion** $w : X_B \to \mathbb{R}$. Dann müssen folgende Bedingungen eingehalten werden:*

(W1) *Wenn $X_{ik} \in X_B$ irrelevant ist, ist $w(X_{ik}) = 0$*

(W2) *$X_{ik} \sim X_{il} \Rightarrow w(X_{ik}) = w(X_{il})$*

(W3) *Sei $F_i \subseteq X_B$ eine Menge von paarweise alternativen Features. Dann gilt $\forall S \subset F_i, S \neq \emptyset$:*

$$\sum_{X_{ik} \in S} w'(X_{ik}) = \sum_{X_{ik} \in F_i} w(X_{ik}) = \hat{w}$$

Dabei ist $w' : (X_B \backslash F_i) \cup S \to \mathbb{R}$ eine Gewichtungsfunktion und $\hat{w}$ das "prototypische" Gewicht für die alternativen Features.

(W4) *Sei AF eine Menge von Features mit*

$$\forall X_{ik} \in AF : X_{ik} \in IF_i \oplus \exists X_{il} \in X_B : X_{ik} \sim X_{il}$$

Dann gilt

$$\forall X_{il} \in X_B : \nexists X_{ik} \in AF : X_{il} \sim X_{ik} \wedge w'(X_{il}) = w(X_{il})$$

wobei w' eine Gewichtungsfunktion für

$$X'_B = X_B \cup AF$$

ist.

Die Bedingungen (W1) und (W2) formalisieren lediglich die naheliegende Behandlung der weiter oben definierten Eigenschaften der Irrelevanz und der Alternativität. Irrelevante Features sollen mit einem Gewicht von 0 belegt werden, zwei zueinander alternative Features sollen das gleiche Gewicht erhalten. (W3) stellt sicher, dass eine Menge von Features, die alle zu ein und demselben Base Feature X_{ik} alternativ sind, in der Summe immer nur mit demselben Gewicht belegt wird. Dieses Gewicht ist gerade $\hat{w}$, das X_{ik} erhalten würde, wenn es ohne die Menge seiner Alternativen gewichtet würde. Dabei spielt

es keine Rolle, wieviele Features die Alternativenmenge enthält. Dadurch wird sichergestellt, dass kein Feature durch wiederholtes Hinzufügen seiner Alternativen überbewertet wird. Schließlich bestimmt (W4), dass sich die Gewichtung eines Basisfeatures nur dann durch Hinzufügen eines neuen Features ändern darf, wenn dieses nicht irrelevant und gleichzeitig alternativ zu einem Basisfeature ist.

Da die Bedingung (W4) relativ komplex formuliert ist, soll als nächstes eine vereinfachte Anforderung (W4′) ausgeführt werden, die die Irrelevanz von Features nicht weiter berücksichtigt:

(W4′) Sei $X_{ik} \notin X_B$ ein neues Feature mit $\exists X_{il} \in X_B : X_{ik} \sim X_{il}$ und w' eine Gewichtungsfunktion für $X_B \cup \{X_{ik}\}$. Dann gilt:

$$\forall X_{im} \in X_B : w'(X_{im}) \neq w(X_{im}) \Rightarrow X_{ik} \sim X_{im}$$

3.2. Aussagen über konkrete Gewichtungsmethoden

In diesem Abschnitt wird für verschiedene Gewichtungsmethoden überprüft, ob sie den Bedingungen (W1) bis (W4) genügen. Es stellt sich heraus, dass ganze Klassen von Gewichtungsmethoden dies nicht tun und damit für den Case Base Ansatz ungeeignet sind.

Lemma 3.2.1 *Keine Methode zur Feature Selection kann die Bedingungen (W1) bis (W4) erfüllen.*

Beweis: Feature Selection stellt immer eine binäre Gewichtungsfunktion dar, d.h. $w(X_k) \in \{0, 1\}$. Es sei t eine Lernaufgabe mit einem Satz an Base Features X_B, von dem sinnvollerweise angenommen werden soll, dass er mindestens ein für t relevantes Feature enthält. X_k sei ein solches Feature, dass mit $w(X_k) = 1$ belegt wird. Wenn jetzt ein neues Feature X_l mit $X_k \sim X_l$ zu den Base Features hinzugefügt wird (also $X'_B = X_B \cup X_l$ und $w' : X'_B \to \{0, 1\}$), so können folgende Fälle eintreten:

- **Beide selektiert:** $w(X_k) = w'(X_k) = w'(X_l) = 1$, was gegen (W3) verstoßen würde, da $w'(X_k) + w'(X_l) = 2 \neq w(X_k)$.
- **Eines selektiert:** $w'(X_k) \neq w'(X_l)$, was ein Widerspruch zu (W2) wäre.
- **Keines selektiert:** $w'(X_k) = w'(X_l) = 0$, was ebenfalls (W3) widerspräche, da diesmal $w'(X_k) + w'(X_l) = 0 \neq w(X_k)$ wäre. □

Dieses Ergebnis schließt Methoden wie PCA [Dunteman 1989], Principal Feature Analysis [Cohen et al. 2002] oder evolutionäre Feature Selection [Morariu et al. 2006] von der Verwendung im Rahmen des Case Base Ansatzes aus.

Lemma 3.2.2 *Keine Methode zur Feature Gewichtung, die $w(X_k)$ unabhängig von $X_B \backslash X_k$ berechnet, kann die Bedingungen (W1) bis (W4) erfüllen.*

Beweis: Sei t wiederum eine Lernaufgabe mit einem Satz an Base Features X_B ohne irrelevante und alternative Features. $X'_B = X_B \cup X_k$ mit $\exists X_l \in X_B : X_k \sim X_l$ sei ein um ein alternatives Feature erweiterer Featuresatz, der mit $w' : X'_B \to \mathbb{R}$ gewichtet wird.

Wenn die Gewichtung der einzelnen Features nun unabhängig voneinander geschieht, so verändert das Hinzufügen von X_k das Gewicht der anderen Features nicht, d.h. es gilt $w(X_l) = w'(X_l)$ Wegen (W2) muss dann $w'(X_k) = w'(X_l) = w(X_l)$ gelten, was jedoch im Widerspruch zu (W3) steht, da $w(X_l) \neq 0$ ist, was durch den Ausschluss irrelevanter Features aus X_B garantiert ist.

□

Dieses Lemma schließt bereits die Gewichtungsmethoden Information Gain [Quinlan 1986] und Relief [Kira und Rendell 1992] aus. Nach diesen beiden Negativaussagen soll im Folgenden gezeigt werden, dass mit der Support Vector Machine durchaus eine Gewichtungsmethode existiert, die die genannten Bedingungen erfüllt. Support Vector Machines basieren auf Vapniks Ergebnissen zur statistischen Lerntheorie [Vapnik 1995]. Sie minimieren im Gegensatz zu vielen anderen Lernverfahren nicht nur das empirische Risiko (also den Fehler auf den Trainingsdatensätzen), sondern ebenfalls das strukturelle Risiko. Dadurch können SVMs verhindern, dass sie durch Überanpassung des Modells auf den Trainingsdatensatz diesen auswendig lernen, was die Vorhersagequalität auf neuen Datensätzen dramatisch verschlechtern würde. Des weiteren bieten SVMs die Möglichkeit, nicht nur lineare Zusammenhänge zwischen den Features zu erlernen. Durch die Verwendung von Kerneln ist es effizient möglich, die Features durch beliebige positiv semi-definite Funktionen zu verknüpfen [Smola und Schölkopf 2003]. Dabei werden die usprünglichen Eingabedaten in einen höherdimensionalen Kernelraum transformiert und dort wird dann die zu lernende Funktion berechnet. Da beim Case Base Ansatz jedoch wie schon erwähnt die Nichtlinearität in den konstruierten Features steckt, genügt es zunächst, die SVM in ihrer einfachsten linearen Form zu benutzen. Die SVM berechnet eine lineare Funktion der Form $w_1X_1 + w_2X_2 + \cdots + w_nX_n$. Im einfachsten Fall, unter Benutzung des Skalarproduktkernels, stellen die X_i die urspünglichen Features dar, falls andere komplexere Kernelfunktionen zum Einsatz kommen, entsprechen die X_i letztendlich zusammengesetzten Features. Im Falle der Klassifizierung ist diese Funktion als maximal trennende Hyperebene zwischen den Instanzen der beiden Klassen zu interpretieren, im Falle der Funktionsregression stellt sie einfach die gesuchte Funktion dar. [Mierswa und Wurst 2005b] benutzen die w_i dieser Funktion nun als Gewichtung der Features im Sinne eines Relevanzmaßes.

Lemma 3.2.3 *Die Featuregewichtung mit Hilfe einer SVM mit linearer Kernelfunktion erfüllt die Bedingungen (W1)-(W4).*

Beweis:

(W1) Es sei vorausgesetzt, dass die SVM tatsächlich die optimale Hyperebene findet. Das bedeutet, dass sowohl der empirische Fehler auf den Trainingsdaten als auch die Länge des Gewichtsvektors w minimiert wird. Das Gewicht w_k für ein Attribut X_k, dass nicht mit der Ergebnisvariable linear korreliert ist, ändert nichts am empirischen Fehler. Wenn die SVM jedoch $w_k > 0$ wählte, verstieße das gegen die angestrebte Minimierung des Gewichtsvektors. An dieser Stelle sollte beachtet werden, dass diese Feststellung nur für lineare Korrelation gilt. Ein Attribut kann durchaus einen komplexeren Zusammenhang mit der Ergebnisvariablen aufweisen und eine lineare SVM könnte dem Attribut ggfs. trotzdem nur ein Gewicht von 0 zuweisen. Hier wird also der Begriff der Irrelevanz auf

lineare Korrelation (bzw. das Fehlen derselben) eingeschränkt. Es bleibt zu untersuchen, ob das praktische Auswirkungen hat.

(W2) Wie schon erwähnt, wird bei der SVM-Optimierung der Gewichtsvektor minimiert, was als

$$w_1^2 + \cdots + w_i^2 + \cdots + w_m^2 \stackrel{!}{=} min$$

notiert werden kann. Mittels der noch zu beweisenden Bedingung (W3) wird w_i wie folgt ersetzt:

$$w_1^2 + \cdots + (\hat{w} - \sum_{j \neq i} w_j)^2 + \cdots + w_m^2 \stackrel{!}{=} min$$

Um nun das Minimum zu finden, muss diese Gleichung für alle Gewichte w_k partiell abgeleitet werden:

$$\frac{\partial}{\partial w_k}(\cdots + (\hat{w} - \sum_{j \neq i} w_j)^2 + w_k^2 + \cdots) = 0$$

$$\Leftrightarrow \; 2w_k - 2(\hat{w} - \sum_{j \neq i} w_j) = 0 \; \Leftrightarrow \; w_k + \sum_{j \neq i} w_j = \hat{w}$$

Die Summe in der letzten Gleichung enthält für jede Ableitung ein weiteres w_k, so dass man auf ein lineares Gleichungssystem der folgenden Form kommt:

$$\vdots$$

$$\cdots + 1w_{i-1} + 0w_i + 1w_{i+1} + \cdots + 1w_{k-1} + 2w_k + 1w_{k+1} + \cdots = \hat{w}$$

$$\vdots$$

Der Nullkoeffizient bleibt dabei immer an der Stelle i, der Koeffizient 2 wandert über alle Stellen mit Außnahme der i-ten Stelle. Ein Beispiel mit drei Attributen X_1, X_2 und X_3, bei dem X_2 und X_3 zueinander alternativ sind, könnte also folgendermaßen aussehen:

$$0w_1 + 1w_2 + 2w_3 = \hat{w}$$
$$0w_1 + 2w_2 + 1w_3 = \hat{w}$$

Löst man dieses Gleichungssystem auf, erhält man $w_2 = w_3$.

Ein alternativer Beweis für (W2) orientiert sich näher an den Optimierungskriterien der SVM. Es muss hierbei vorausgesetzt werden, dass sich alternative Features durch Normalisierung in identische Features umwandeln lassen. Da dies jedoch in den meisten SVM-Implementierungen so gehandhabt wird, stellt diese Vorraussetzung keine allzu große Einschränkung an die Allgemeingültigkeit des Beweises dar. Aus der Minimierung der Langrangefunktion ergibt sich nach der Nullsetzung der Ableitungen u.a. folgende Gleichung zur Bestimmung der Featuregewichte (vgl. [Mierswa 2006]):

$$w_j = \sum_{i=1}^{n} \alpha_i y_i x_{ij}$$

Dabei ist w_j das Gewicht des $j$ten Features, α_i der Langrangefaktor zum $i$ten Trainingsbeispiel, y_i das Label des $i$ten Trainingsbeispiels und x_{ij} der Wert von Feature j im $i$ten

Trainingsbeispiel. Laut Vorraussetzung gilt nach der Normalisierung $\forall i : x_{ij} = x_{ik}$ falls $X_j \sim X_k$. Dann gilt folgende Identität:

$$w_j = \sum_{i=1}^{n} \alpha_i y_i x_{ij} = \sum_{i=1}^{n} \alpha_i y_i x_{ik} = w_k$$

Damit ist wiederum gezeigt, dass alternative Features das gleiche Gewicht erhalten.

(W3) Diese Bedingung sagt aus, dass sich der Gewichtsvektor nicht durch Hinzufügen von alternativen Features ändert. Zum Zwecke dieses Beweises sei wiederum angenommen, dass sich alternative Features durch Normalisierung in identische Features umwandeln lassen. Die von einer klassifizierenden SVM zu lernende optimale Trennungshyperebene mit Gewichtsvektor w kann als

$$y = sign(w_1 x_1 + \cdots + w_i x_i + \cdots + w_m x_m + b)$$

notiert werden. Nun muss gezeigt werden, dass der Gewichtsvektor nicht verändert wird, wenn man das gleiche Feature mehrmals hinzufügt. Das $k-1$-fache Hinzufügen von alternativen Features ergibt

$$y = sign(w_1 x_1 + \cdots + (w_i^1 + \cdots + w_i^k) x_i + \cdots + w_m x_m + b)$$

Die optimale Hyperebene, welche die SVM berechnet, ändert sich dadurch jedoch nicht. Dies bedeutet zum einen, dass sich die anderen Gewichte w_j nicht ändern, zum anderen, dass $w_i = \sum_{l=1}^{k} w_i^l$ gilt. Dies beweist Bedingung (W3).

(W4) Wieder vorrausgesetzt, dass die SVM mit ausreichend Trainingsdatensätzen die optimale Hyperebene findet, ist durch (W1) sichergestellt, dass das Hinzufügen von irrelevanten Features den Gewichtsvektor nicht ändert. Die Bedingungen (W2) und (W3) besagen zudem, dass alternative Features die Hyperebene ebenfalls nicht beeinflussen.

□

4. Ähnlichkeit von Lernaufgaben

Nachdem im vorherigen Kapitel 3 die Grundlagen zur Gewichtsbestimmung mittels der SVM erläutert wurden, geht es jetzt um die Bestimmung der Ähnlichkeit zwischen zwei Lernaufgaben auf Basis ebendieser Gewichtung.

4.1. Abstandsmaß über die Base Features

Das Abstandsmaß, mit dessen Hilfe für eine neue Lernaufgabe t die k ähnlichsten Lernaufgaben t_i aus der Case Base bestimmt werden, sollte laut [Mierswa und Wurst 2005b] folgende Anforderungen erfüllen:

Bedingungen für die Distanz 1 *Ein* ***Distanzmaß*** d *für Lernaufgaben ist eine Zuordnung* $d : T \times T \rightarrow \mathbb{R}^+$ *die folgenden Bedingungen genügen muss:*

(D1) $d(t_1, t_2) = 0 \Leftrightarrow t_1 = t_2$

(D2) $d(t_1, t_2) = d(t_2, t_1)$

(D3) $d(t_1, t_3) \leq d(t_1, t_2) + d(t_2, t_3)$

(D4) $d(t_1, t_2) = d(t'_1, t'_2)$ *falls* $X'_B = X_B \cup IF$ *und* $IF \subseteq IF_1 \cap IF_2$

(D5) $d(t_1, t_2) = d(t'_1, t'_2)$ *falls* $X'_B = X_B \cup AF$ *und* $\forall X_k \in AF : \exists X_l \in X_B : X_k \sim X_l$

(D1)-(D3) stellen die Bedingungen für eine Metrik dar. Diese Eigenschaften sind wichtig für die Indizierung der einzelnen Lernaufgaben in der Case Base sowie für die effiziente Suche nach ihnen. Als Mittel der Wahl für diese Aufgabe wurden M-Trees vorgeschlagen, siehe auch [Ciaccia et al. 1997]. Bedingung (D4) verhindert, dass das Hinzufügen von irrelevanten Features zu den Lernaufgaben den Abstand zwischen ihnen verändert, während (D5) das gleiche für alternative Features gewährleistet. Die letzten beiden Regeln mögen überflüssig erscheinen, da die beiden Probleme Irrelevanz und Alternativität bereits durch die Gewichtungsbedingungen (W1)-(W4) behandelt wurden. Durch die gestellten Anforderungen an das Distanzmaß d kann es jedoch unabhängig von einer bestimmten Gewichtungsfunktion als Lieferant für eine geeignete Indizierung als Ähnlichkeitsmaß für Lernaufgaben auf Featurebasis dienen.

4.2. Betrachtung verschiedener Metriken

Auch für den Aufgabenbereich der Abstandsmessung zwischen verschiedenen Lernaufgaben sollen nun mehrere Metriken auf ihre Eignung hin überprüft werden.

4.2.1. Minkowski-Metriken

Zunächst werden die von den sogenannten *p-Normen* induzierten Minkowski-Metriken [JAIN et al. 1999, KÜRSTEN 2006] untersucht. Eine Norm stellt die Verallgemeinerung des Längenbegriffes für Vektoren dar. p-Normen (mit $p \in \mathbb{N}$) im endlichdimensionalen $\mathbb{R}^n$ sind definiert als

$$\|x\|_p = \sqrt[p]{\sum_{i=1}^{n} |x_i|^p} \quad , mit\ x \in \mathbb{R}^n$$

Eine durch eine Norm induzierte Metrik ist definiert als

$$d(x, y) = \|x - y\| \quad , mit\ x, y \in \mathbb{R}^n$$

Bekannte Minkowski-Metriken sind die *Manhattandistanz* (p=1) und die *euklidische Distanz* (mit p=2).

Lemma 4.2.1 *Alle Minkowski-Metriken erfüllen die Bedingungen (D1)-(D4).*

Beweis: Da die Bedingungnen (D1)-(D3) durch jede beliebige Metrik erfüllt werden, ist nur noch die Bedingung (D4) zu betrachten:

Es seien zwei Trainingsdatensätze t_i, t_j mit den Basisfeatures X_B gegeben und eine Gewichtungsfunktion w, die die Bedingungen (W1)-(W5) erfüllt. t_i' und t_j' seien jeweils um irrelevante Features X_{iq} bzw. X_{jq} erweitert. Dann gilt:

$$d(t_i', t_j') = \sqrt[p]{\left(\sum_{X_{ip}, X_{jp} \in X_B} (w_i'(X_{ip}) - w_j'(X_{jp}))^p\right) + \underbrace{\sum_{X_{iq}, X_{jq} \in IF} (w_i'(X_{iq}) - w_j'(X_{jq}))^p}_{=0}}$$

$$= d(t_i, t_j)$$

□

Lemma 4.2.2 *Die Manhattandistanz erfüllt die Bedingung (D5).*

Beweis: Es seien wiederum zwei Trainingsdatensätze t_i, t_j mit den Basisfeatures X_B sowie die Gewichtungsfunktion w gegeben. X_B enthalte keine alternativen Features. Nun wird ein alternative Feature X_{ik} zu X_B hinzugefügt, so dass $X_B' = X_B \cup \{X_{ik}\}$ mit $\exists X_{il} \in X_B : X_{il} \sim X_{ik}$ gilt. Aus den Gewichtungsbedingungen (W2) und (W3) kann

$$w'(X_{ik}) = w'(X_{il}) = \frac{w(X_{il})}{2} \quad und \quad w'(X_{jk}) = w'(X_{jl}) = \frac{w(X_{jl})}{2}$$

gefolgert werden, aus (W4) folgt

$$\forall q \neq k : w'(X_{iq}) = w(X_{iq}) \quad und \quad \forall q \neq k : w'(X_{jq}) = w(X_{jq})$$

Dann gilt für jede aus einer p-Norm induzierten Metrik:

$$d(t'_i, t'_j) = \sqrt[p]{S + 2(w'(X_{ik}) - w'(X_{jk}))^p} = \sqrt[p]{S + 2\left(\frac{w(X_{ik})}{2} - \frac{w(X_{jk})}{2}\right)^p}$$

$$= \sqrt[p]{S + \frac{1}{2^{(p-1)}}(w(X_{ik}) - w(X_{jk}))^p} \overset{\text{für p>1}}{\neq} \sqrt[p]{S + (w(X_{ik}) - w(X_{jk}))^p} = d(t_i, t_j)$$

$$mit \quad S = \sum_{q=1, q\neq k}^{|X_B|} (w'(X_{iq}) - w'(X_{jq}))^p = \sum_{q=1, q\neq k}^{|X_B|} (w(X_{iq}) - w(X_{jq}))^p$$

Wie man sieht, muss der Faktor $\frac{1}{2^{(p-1)}}$ den Wert 1 ergeben, um die Gleichheit von $d(t'_i, t'_j)$ und $d(t_i, t_j)$ zu erhalten. Mittels folgender einfacher Gleichungsumformungen ermittelt man den für die Erfüllung von (D5) notwendigen Wert für p

$$\frac{1}{2^{(p-1)}} = 1$$

$$\Leftrightarrow 2^{(p-1)} = 1$$

$$\Leftrightarrow p - 1 = log_2(1)$$

$$\Leftrightarrow p - 1 = 0$$

$$\Leftrightarrow p = 1$$

Die auf der 1-Norm basierende Minkowski-Metrik ist gerade die Manhattandistanz.

□

Korollar 4.2.1 *Die Manhattandistanz ist die einzige Minkowski-Metrik, die die Bedingungen (D1)-(D5) erfüllt.*

Beweis: Diese Aussage ergibt sich unmittelbar aus den Lemmata 4.2.1 und 4.2.2.

□

4.2.2. Quadratische Formen

Quadratische Formen werden in einigen Distanzmaßen für Vektoren in der Statistik benutzt. Dabei wird, im Gegensatz zur euklidischen Distanz, bei der nur das Skalarprodukt des Differenzvektors mit sich selbst gebildet wird, hier eine Matrix in dieser Multiplikation zwischengeschaltet. Im Allgemeinen ergibt sich dabei folgende Notation (siehe [Rencher 1998], S.404):

$$\mathbf{a}^T\mathbf{S}\mathbf{a} = \sum_{i=1}^{n} a_i^2 s_{ii} + \sum_{i\neq j} a_i a_j s_{ij}$$

Dabei ist $\mathbf{a}$ der n-dimensionale Differenzvektor, $\mathbf{a}^T$ der transponierte Differenzvektor und $\mathbf{S}$ eine symmetrische $n \times n$-Matrix. Die Matrix S ermöglicht es zum einen, über die Einträge auf der Hauptdiagonalen die Komponenten des Differenzvektors unterschiedlich zu

gewichten und zum anderen, mittels der übrigen Einträge Beziehungen zwischen verschiedenen Komponenten in das Distanzmaß einfließen zu lassen. Ein Beispiel für ein solches Maß ist die Mahalanobis-Distanz [Mahalanobis 1936], die folgendermaßen definiert ist:

$$d_M(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T \, \Sigma^{-1} \, (\mathbf{x} - \mathbf{y})}$$

Die Matrix Σ^{-1} ist dabei die Inverse der Kovarianzmatrix der beiden Vektoren $\mathbf{x}$ und $\mathbf{y}$. Sie sorgt dafür, dass die einzelnen Variablen unkorreliert sind und die gleiche Varianz haben ([Rencher 1998], S.23). Um zu erläutern, wie die Kovarianzmatrix der Gewichtsvektoren X_k und X_l der Lernaufgaben T_k und T_l bestimmt werden kann, sind einige Definitionen und Zwischenschritte nötig. Zunächst sei

$$\hat{X} = \begin{pmatrix} X_{1k} & \cdots & X_{mk} \\ X_{1l} & \cdots & X_{ml} \end{pmatrix}$$

die Matrix der Featuregewichte der zu vergleichenden Gewichtsvektoren. Die Spalten enthalten die einzelnen Features, die Zeilen repräsentieren die beiden Gewichtsvektoren bzw. die Lernaufgaben. Als nächstes benötigt man die Mittelwerte aller Featuregewichte über die beiden Lernaufgaben:

$$\overline{x_i} = \frac{X_{ik} + X_{il}}{2}$$

Die Mittelwerte fließen in die Berechnung der Einzelkovarianzen zwischen je zwei Features ein. Die Kovarianz zwischen den Features $\hat{X_n}$ und $\hat{X_o}$ ist

$$C(\hat{X_n}\hat{X_o}) = (X_{nk} - \overline{X_n})(X_{ok} - \overline{X_o}) + (X_{nl} - \overline{X_n})(X_{ol} - \overline{X_o})$$

Diese Einzelkovarianzen $C(\hat{X_n}\hat{X_o})$ bilden die Einträge $(\sigma_{n,o})$ der Kovarianzmatrix Σ, die für Lernaufgaben mit m Features die Dimension $m \times m$ hat. Weiterhin ist Σ symmetrisch, da

$$C(\hat{X_n}\hat{X_o}) = C(\hat{X_o}\hat{X_n}).$$

Leider hat sich herausgestellt, dass dieses Distanzmaß für die Zwecke dieser Arbeit nicht geeignet ist. Wenn man, wie im Beweis zu Lemma 4.2.2, versucht, ein zu einem Basisfeature X_i alternatives neues Feature X_j zu beiden Trainingsdatensätzen hinzuzufügen, führt das unter Annahme der Gewichtingsbedingungen (W1)-(W4) dazu, dass die beiden neuen Gewichte $w'(X_i) = w'(X_j) = \frac{1}{2}w(X_i)$ entstehen. Das wiederum führt zu einer Kovarianzmatrix Σ', deren i-te und j-te Zeile und Spalte jeweils gleich sind. Zur Veranschaulichung sollen die folgenden beiden Matrizen dienen. Σ sei die Kovarianzmatrix der beiden Trainingsdatensätze mit n Features ohne Alternativen und Σ' die Kovarianzmatrix, bei der an beiden Datensätzen ein zum n-ten Attribut alternatives Feature angehängt ist.

$$\Sigma = (C_{i,j}) = \begin{pmatrix} C_{11} & \cdots & C_{1n} \\ \vdots & & \vdots \\ C_{n1} & \cdots & C_{nn} \end{pmatrix}$$

$$\Sigma' = (C'_{i,j}) = \begin{pmatrix} C_{11} & \cdots & \frac{1}{2}C_{1n} & \frac{1}{2}C_{1n} \\ \vdots & & \vdots & \vdots \\ \frac{1}{2}C_{n1} & \cdots & \frac{1}{4}C_{nn} & \frac{1}{4}C_{nn} \\ \frac{1}{2}C_{n1} & \cdots & \frac{1}{4}C_{nn} & \frac{1}{4}C_{nn} \end{pmatrix}$$

Durch die Gleichheit der letzten beiden Zeilen- und Spaltenvektoren ist die Invertierbarkeit der Matrix nicht mehr gegeben, wie der folgende Exkurs in die Theorie der Matrizen und linearen Gleichungssysteme nach [Dörfler und Peschek 1988] zeigt: Wenn eine Matrix $\mathbf{A}$ zwei gleiche Zeilen (Spalten) enthält, so kann eine dieser beiden Zeilen (Spalten) durch eine elementare Matrixumformung (siehe [Dörfler und Peschek 1988], S.229f) in eine Nullzeile (-spalte) umgeformt werden. Daraus folgt, dass für die Determinate einer solchen Matrix $det(\mathbf{A}) = 0$ gilt (Satz 10.4.3). Dies wiederum bedeutet, dass $\mathbf{A}$ singulär und damit nicht invertierbar ist. Somit ist die Mahalanobis-Distanz für Lernaufgaben mit alternativen Features nicht definiert.

Auch bei Verwendung einer quadratischen Form ohne invertierte Matrix wird in der Regel immer das gleiche Problem wie bei der euklidischen Distanz auftreten, nämlich die Unverträglichkeit der Quadrierung mit der Gewichtsaufteilung auf mehrere alternative Features (vgl. wiederum Beweis zu Lemma 4.2.2). Man könnte zwar vermutlich ein Regelwerk erstellen, welches bei Vorkommen von mehreren gleichen Gewichten die entsprechenden Matrixeinträge skaliert, so dass die Bedingung (D5) von einem solchermaßen konstruierten Distanzmaß erfüllt wird, doch das führte auf die nachträgliche Erkennung von alternativen Features nur anhand gleicher Gewichte hinaus. Der Schluß von gleichen Featuregewichten auf die Alternativität der entsprechenden Features kann jedoch im Allgemeinen nicht gezogen werden, da in den Gewichtungsbedingungen nirgendwo eine entsprechende Forderung enthalten war ((W2) fordert gerade die Umkehrung).

4.2.3. Weitere Abstandsmaße

Da es zur Evaluierung des Case Base Ansatzes sinnvoll erscheint, neben der Manhattandistanz zumindest noch ein weiteres Maß zur Verfügung zu haben, welches die Bedingungen (D1)-(D5) erfüllt, werden im Weiteren die auf reelwertige Vektoren anwendbaren Distanz- und Ähnlichkeitsmaße aus dem Clustering-Plugin der Lernumgebung YALE [Mierswa et al. 2006] betrachtet. Da sich bis jetzt stets die Bedingung (D5) als diejenige herausgestellt hat, welche ein neues Distanzmaß nicht erfüllt, wird sie in einem automatisierten Test für jedes Maß überprüft. Zu diesem Zweck wird zunächst der Abstand von zwei Zufallsvektoren der Länge n ermittelt. Dann wird der Wert in der i-ten Stelle jedes Vektors halbiert und derselbe halbierte Wert als $n + 1$-te Stelle an die neuen Vektoren angehängt. Auch von dem neuen Vektorpaar wird der Abstand (bzw. die Ähnlichkeit) berechnet und dieser mit dem Abstand der ursprünglichen Vektoren verglichen. Dieser Test simuliert das Hinzufügen eines zu einem Basisfeature alternativen neuen Features zu einer Lernaufgabe. Falls sich der Abstand dabei verändert, kann mit Sicherheit festgestellt werden, dass das getestete Maß den Anforderungen nicht entspricht. Auf diese Weise konnte eine Reihe von Maßen als nicht geeignet identifiziert werden. Der Vollständigkeit halber werden diese Maße trotzdem kurz vorgestellt. $\mathbf{x}$ und $\mathbf{y}$ seien im Folgenden jeweils n-dimensionale Vektoren mit den Komponenten x_i und y_i.

Canberra-Distanz

Die Canberra-Distanz berechnet sich nach folgender Formel:

$$d_{Can}(\mathbf{x}, \mathbf{y}) = 1 - \sum_{i=1}^{n} \frac{|x_i - y_i|}{x_i + y_i}$$

Sie bezieht nicht nur den relativen Abstand zwischen zwei Punkten mit ein, sondern auch die Entfernung zum Ursprung. [Emran und Ye 2001] benutzen dieses Distanzmaß für ihre Arbeit im Bereich Intrusion Detection, um Aktivitäten zu erkennen, die von normalen Abläufen in einem Computersystem abweichen. Für den Case Base-Ansatz eignet es sich leider aus mehreren Gründen nicht. Zum einen ergeben sich bei dem oben beschriebenen Test zur Überprüfung der Bedingung (D5) Abweichungen beim Einbringen der alternativen Features, zum anderen kann man schon an der Formel erkennen, dass das Hinzufügen von irrelevanten Features dazu führt, dass die Canberra-Distanz nicht mehr berechnet werden kann, da der Nenner $x_i + y_i$ in diesem Fall zu null wird.

Cosinus-Ähnlichkeitsmaß

Bei diesem Maß werden $\mathbf{x}$ und $\mathbf{y}$ als Richtungsvektoren aufgefasst, die einen Winkel α einschließen. Der Kosinus dieses Winkels wird als Maßzahl für die Ähnlichkeit genutzt. Der Wert von $\cos(\alpha)$ wird folgendermaßen berechnet:

$$s_{Cos}(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{n} x_i * y_i}{\sqrt{\sum_{i=1}^{n} {x_i}^2 * \sum_{i=1}^{n} {y_i}^2}}$$

Korrelationskoeffizient

Der Korrelationskoeffizient ähnelt dem Cosinus-Ähnlichkeitsmaß, nur werden die Vektorkomponenten mit ihren jeweiligen Mittelwerten standardisiert.

$$s_{Cor}(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{n} (x_i - \overline{x}) * (y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \overline{x})^2 * \sum_{i=1}^{n} (y_i - \overline{y})^2}}$$

mit

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} (x_i)$$

und

$$\overline{y} = \frac{1}{n} \sum_{i=1}^{n} (y_i)$$

Ebenso wie das Cosinus-Ähnlichkeitsmaß eignet sich auch der Korrelationskoeffizient nicht für den Case Base-Ansatz, wie mit Hilfe des oben beschriebenen Testverfahrens gezeigt werden kann.

Dice-Ähnlichkeitsmaß

Der Dice Similarity Coefficient (DSC) nach [Dice 1945] wurde ursprünglich dazu verwendet, die Überlappung zweier Vektoren mit nominalen Attributen zu messen. Er wird in dieser Form u.a. im String Matching angewendet, siehe etwa [McEnery et al. 1994]. Folgende Formel, die jedoch leider (D5) ebenfalls nicht erfüllt, stellt die Abwandlung für reelwertige Vektoren dar:

$$s_{Dice}(\mathbf{x}, \mathbf{y}) = \frac{2 * \sum_{i=1}^{n} x_i * y_i}{\sum_{i=1}^{n} x_i + y_i}$$

Dynamic Time Warping

Dynamic Time Warping nach [Myers und Rabiner 1981] wird klassischerweise für eindimensionale Zeitreihen eingesetzt. Die Besonderheit der DTW-Ansatzes besteht darin, dass die Zuordnung der Komponenten der beiden zu vergleichenden Zeitreihen bzw. Vektoren $\mathbf{x}$ und $\mathbf{y}$ nicht fest vorgegeben ist. Vielmehr wird beim Dynamic Time Warping in Betracht gezogen, dass sich die Zeitpunkte verschoben haben können oder eine Zeitreihe im Vergleich zur anderen gedehnt sein könnte, z.B. durch unterschiedliche Geschwindigkeiten bei zwei ansonsten ähnlichen Vorgängen. Die einzige feste Bedingung für die Zuordnungsfunktion der Komponenten der beiden Vektoren ist die Monotonie. Das bedeutet, es ist zwar möglich, z.B. x_1 mit y_2 und x_2 mit y_4 zu verknüpfen, dann darf aber x_3 nicht mehr y_3 zugeordnet werden. Unter den $O(n^2)$ vielen möglichen Zuordnung wird mittels dynamischer Programmierung diejenige ausgewählt, bei der die Summe der Differenzen der einander zugeordneten Komponenten minimal ist. Der wiederholte Test mit Zufallsvektoren ergibt, dass auch dieses Maß nicht für den Einsatz im Case Base-Ansatz geeignet ist.

Skalarprodukt

Auch das Skalarprodukt, das sich nach der Formel

$$d_{scal}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} x_i * y_i$$

berechnet, eignet sich nicht für den Case Base-Ansatz, da es die Bedingung (D5) nicht erfüllt, wie sich leicht zeigen läßt.

Jaccard Index

Der Jaccard Index (auch Jaccard Koeffizient) wird in seiner ursprünglichen Form dazu benutzt, die Ähnlichkeit zweier Mengen zu bestimmen. Er ist definiert als die Größe der Schnittmenge geteilt durch die Größe der Vereinigung der beiden zu vergleichenden Mengen. Um in auch auf reele Attributvektoren, wie sie im Case Base-Ansatz benutzt werden, anwenden zu können, wurde in [Mierswa et al. 2006] folgende Formel verwendet:

$$d_{jaccard}(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{n} x_i * y_i}{\sum_{i=1}^{n} x_i + y_i - \sum_{i=1}^{n} x_i * y_i}$$

Auch für den Jaccard Index stellt sich heraus, dass er Bedingung (D5) nicht erfüllt.

Produktmaximum

Das Produkmaximum bildet, wie der Name sagt, das Maximum über die Produkte der Einzelkomponenten:

$$d_{maxProduct}(\mathbf{x}, \mathbf{y}) = \max_i(x_i * y_i)$$

Eben diese Produktbildung führt jedoch, ähnlich wie die Quadrierung bei der Euklidischen Distanz wiederum dazu, dass bei Hinzufügen von alternativen Attributen der Abstand verändert wird. Diese Eigenschaft verstößt gegen (D5).

4.2.4. Overlap-Distanz

Die Overlap-Distanz ist außer der Manhattandistanz das einzige Maß aus dem Clustering Plugin von YALE, welches den oben beschriebenen automatisierten Test der Bedingung (D5) besteht. Daher wird die Overlap-Distanz, die sich nach der Formel

$$d_{overlap}(x, y) = 1 - \frac{\sum_{i=1}^{n} min\,(x_i, y_i)}{min\,(\sum_{i=1}^{n} x_i, \sum_{i=1}^{n} y_i)}, \quad mit \quad x, y \in \mathbb{R}^n$$

berechnet, weiter theoretisch untersucht. Wie sich dabei leider herausstellt, stellt dieses Maß keine Metrik dar:

Lemma 4.2.3 *Die Bedingung (D1) wird von der Overlap-Distanz nicht erfüllt.*

Beweis: Bedingung (D1) verlangt, dass zwei Vektoren nur dann einen Abstand von 0 haben, wenn die beiden Vektoren identisch sind. Formal muss also $d(\mathbf{x}, \mathbf{y}) = 0 \Rightarrow \mathbf{x} = \mathbf{y}$ gelten. Falls man die Overlap-Distanz für zwei Vektoren $\mathbf{x}, \mathbf{y}$ berechnet, die komponentenweise geordnet sind ($\forall i : x_i < y_i$), so erhält man ebenfalls den Wert 0, was der Anforderung (D1) widerspricht.

□

Diese Tatsache stellt jedoch nicht unbedingt ein Ausschlusskriterium dar. Die Voraussetzung, dass das gesuchte Distanzmaß eine Metrik sein muss, wird lediglich für eine effiziente Indizierbarkeit der Fallbasis benötigt. Diese Eigenschaft ist aber zumindest für erste Experimente nicht zwingend erforderlich, sofern die Anzahl der Lernaufgaben in der Fallbasis nicht zu groß wird.

Lemma 4.2.4 *Die Overlap-Distanz erfüllt die Bedingungen (D4) und (D5).*

Beweis: (D4) Es seien zwei n-dimensionale Gewichtsvektoren $\mathbf{x}, \mathbf{y}$ gegeben, welche mit Hilfe einer Gewichtungsfunktion w, die die Bedingungen (W1)-(W5) erfüllt, aus zwei Trainingsdatensätzen t_x und t_y gewonnen wurden. t'_x und t'_y seien jeweils um irrelevante Features x_{n+1} bzw. y_{n+1} erweitert. Dann gilt:

$$d_{overlap}(t'_x, t'_y) = \frac{\sum_{i=1}^{n} \min(x_i, y_i) + 0}{\min\,(\sum_{i=1}^{n} x_i + 0, \sum_{i=1}^{n} y_i + 0)} = d_{overlap}(t_x, t_y)$$

(D5) Es seien die gleichen Voraussetzungen wie oben gegeben. Jetzt wird zu t_x und t_y jeweils ein alternatives Feature hinzugefügt x_{n+1} bzw. y_{n+1}. ObdA seien diese neuen

Features alternativ zu x_n bzw. y_n, so dass sich folgende neuen Gewichtungen ergeben: $x'_n = x'_{n+1} = \frac{1}{2}x_n$ und $y'_n = y'_{n+1} = \frac{1}{2}y_n$. Dann gilt:

$$d_{overlap}(t'_x, t'_y) = \frac{\sum_{i=1}^{n-1} \min(x_i, y_i) + 2 * \min(\frac{1}{2}x_n, \frac{1}{2}x_n)}{\min\left(\sum_{i=1}^{n-1} x_i + 2 * \frac{1}{2}x_n, \sum_{i=1}^{n-1} y_i + 2 * \frac{1}{2}y_n\right)} = d_{overlap}(t_x, t_y)$$

□

Da die Overlap-Distanz also zumindest teilweise für den Case Base-Ansatz geeignet erscheint, wird sie als zur Manhattan-Metrik alternatives Distanzmaß in den Experimenten eingesetzt werden.

5. Weiterführende Ansätze zur Ähnlichkeitsbestimmung

Dieses Kapitel befasst sich mit der in [Mierswa und Wurst 2005b] beschriebenen Erweiterung des Grundansatzes zur Suche von ähnlichen Problemen, welcher im vorherigen Kapitel ausführlich erläutert wurde.

5.1. Erweiterung des Abstandmaßes auf konstruierte Attribute

Die bisher beschriebene Grundversion des Case Base-Ansatzes hat den Nachteil, dass sie nur einmal pro Problem angewendet werden kann. Da sich die Basisgewichte eines Problemes bei gleichbleibender Gewichtungsfunktion niemals ändern, würden bei wiederholten Anfragen an die Case Base immer die gleichen Fälle und somit auch die gleichen konstruierten Features zurückgegeben werden. Weitere Anfragen würden also keinen neuen Beitrag zur Lösung des aktuellen Problems liefern. Um dennoch mehrmals während eines Feature Generator-Laufes eine Anfrage an die Case Base stellen zu können, erweitern [Mierswa und Wurst 2005b] das gewichtungsbasierte Ähnlichkeitsmaß um Informationen, die sich während des Laufes ergeben. Im Verlauf des Feature Generation Algorithmus werden für das aktuelle Problem Merkmale konstruiert, die sich als gut erweisen bzw. zu diesem Problem passen. Von dieser Beobachtung ausgehend bietet es sich an, ebendiese konstruierten Features in den Vergleich mit einzubeziehen. Im Weiteren werden zunächst zwei Methoden zum Vergleich konstruierter Features betrachtet, bevor es anschließend um die Kombination von gewichtungsbasiertem und konstruktionsbasiertem Abstand zwischen Lernproblemen geht.

5.1.1. Syntaxbasierter Merkmalsvergleich

Wie schon in Abschnitt 2.3 beschrieben lassen sich konstruierte Features als Funktions(teil)bäume darstellen. Die Blätter dieser Funktionsbäume bestehen aus Basisattributen und Konstanten, die inneren Knoten und die Wurzel aus funktionalen Verknüpfungen (siehe Abbildung 5.1). Wenn man nun die Ähnlichkeit zweier solcher Funktionsbäume bestimmen will, könnte zunächst auf den Gedanken kommen, dies syntaxbasiert, also etwa mit Hilfe eine Graph Matching Algorithmus zu versuchen. Wie bereits [Richardson 1968] bewiesen hat, ist der Vergleich zweier Funktionen, die aus einer kleinen Menge an Grundfunktionen zusammengesetzt sind, nicht lösbar.
[Zhang et al. 1995] haben gezeigt, dass dieses Problem selbst bei Einschränkung auf ungeordnete, annotierte Bäume immerhin noch NP-vollständig ist. Dies ist jedoch nicht der einzige Grund, aus dem auf syntaxbasierte Vergleichsansätze hier nicht näher eingegangen

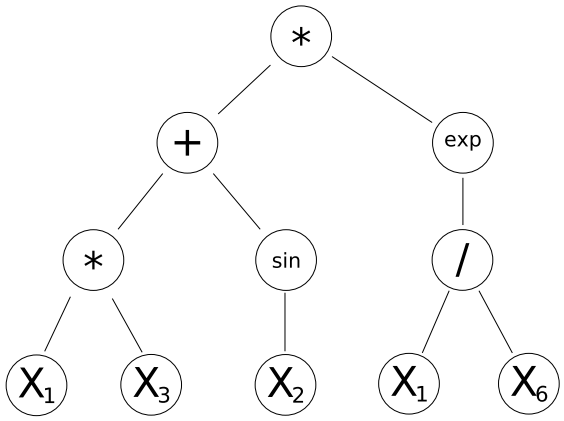
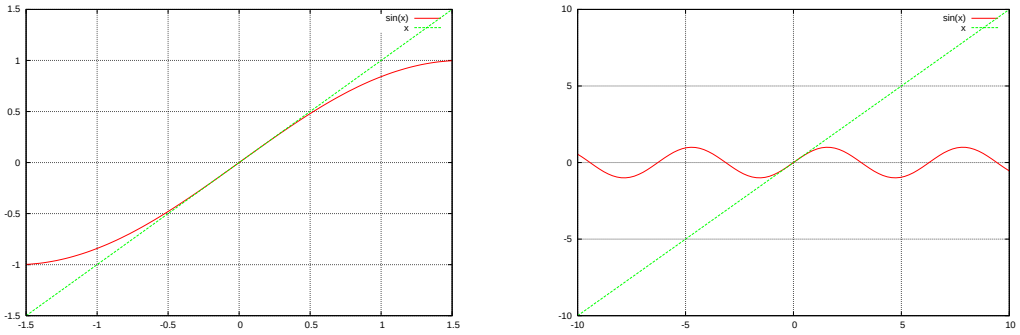


Abbildung 5.1.: Konstruiertes Feature $((x_1 * x_3) + (\sin(x_2))) * \left(e^{(\frac{x_1}{x_6})}\right)$

wird. Viel graviender ist das Problem, dass die Verteilung der Trainingsdaten bei diesen Ansätzen in keinster Weise mit einbezogen wird. Diese Information kann jedoch für den Case Base-Ansatz von erheblicher praktischer Bedeutung sein. Als Beispiel hierfür soll der Vergleich zwischen den beiden Funktionen $f(x) = x$ und $g(x) = \sin(x)$ herangezogen werden. Betrachtet man einen Wertebereich für x von $[-\frac{1}{2}, \frac{1}{2}]$ (Abb. 5.2(a)), so erscheinen die beiden Funktionen beinahe identisch. Legt man jedoch einen Wertebereich von $[-10, 10]$ zu Grunde (Abb. 5.2(a)), so zeigen sich natürlich gravierende Unterschiede im Funktionsverlauf.



(a) Wertebereich (-1.5, 1.5)

(b) Wertebereich (-10, 10)

Abbildung 5.2.: Vergleich von $f(x) = x$ und $g(x) = \sin x$ in verschiedenen Wertebereichen

5.1.2. Samplingbasierter Merkmalsvergleich

Um den oben erläuterten Problemen zu entgehen, bietet sich der Vergleich konstruierter Features auf Basis von Stichproben an [Mierswa und Wurst 2005b]. Dabei wird für die

Basisattribute eine kleine Menge von Stichprobenwerten gemäß der jeweiligen Verteilung in den Trainigsdaten zufällig erzeugt. Wie in Abb. 5.1 deutlich wird, ist es dann möglich, diese künstlich erzeugten Werte in ein zusammengesetztes Attribut einzusetzten und ein Ergebnis zu berechnen. Anhand dieser Ergebnisse kann dann z.B. die Korrelation zwischen zwei Features bestimmt werden.

Der Gesamtablauf des samplingbasierten Vergleiches eines aktuellen Lernproblemes t_i mit einem bereits gelösten Problem t_j aus der Case Base sieht folgendermaßen aus: Zunächst wird in einer äußeren Schleife für jedes konstruierte Feature X_{ik} des neuen Problems dasjenige Feature X_{jl} des alten Falles gesucht, welches den kleinsten Einzelabstand zu X_{ik} hat. Diese Einzelabstände werden aufsummiert und ergeben schließlich den samplingbasierten Gesamtabstand zwischen t_i und t_j. Diese Vorgehensweise, nämlich dass gerade über die Features des neuen Falls zuerst iteriert wird, bietet sich aus folgendem Grund an: Die Anfrage an die Case Base und damit der hier beschriebene Vergleich wird i.d.R. zu einem Zeitpunkt des Feature-Generation-Laufes stattfinden, zu dem für den neuen Fall noch nicht viele gute Features konstruiert wurden. Würde umgekehrt zuerst über die größere Featuremenge des alten Problems iteriert, so würde sich der Abstand unnötig vergrößern, wenn für Features des alten Problems noch keine korrespondierenden Features im neuen Fall existieren.

Wie sich der Einzelabstand zwischen zwei Features bestimmt, wurde oben schon kurz umrissen. Da das Ziehen der Stichprobe für ein Basisattribut auf Grundlage seiner Verteilung in den Trainingsdaten geschieht, die Trainingsdaten aber zumindest für den bereits in der Case Base abgelegten Fall nicht mehr zur Verfügung stehen, müssen jetzt zusätzlich zu den Gewichten der Basisattribute Kennzahlen zur ihrer Verteilung wie z.B. Mittelwert und Standardabweichung in der Case Base gespeichert werden. Es ist zu überlegen, wie genau die möglicherweise stark unterschiedlichen Verteilungen auf den Basisattributen in die Stichprobenziehung einfließen sollten. Eine Möglichkeit ist, lediglich gemäß der Verteilung im Trainingsdatensatz des neuen Falles zu ziehen. Ein Argument für diese Möglichkeit ist die Wichtigkeit des aktuellen Problems. Wenn ein konstruiertes Feature aus einem alten Fall in den Feature-Generator-Lauf mit einfließt, wird es dort letztendlich nur aufgrund der Trainingsdaten des aktuellen Lernproblems benutzt und evaluiert, und nicht auf Basis der Verteilung der Trainingsdaten in seinem alten Fall. Eine andere Möglichkeit besteht darin, einen Teil des Samplings gemäß der Verteilung im neuen Trainingsdatensatz durchzuführen und einen Teil gemäß der Verteilung im alten Problem. Diese zweite Vorgehensweise stellt einen vielleicht einen etwas ausgewogeneren Kompromiss zwischen der alten und der neuen Verteilung dar.

5.2. Zweiphasenansatz

Die im letzten Abschnitt vorgestellte Erweiterung des Grundansatzes zur Bestimmung der Ähnlichkeit von Lernproblemen bringt für den Case Base-Ansatz zwei Vorteile. Der erste wurde schon genannt: Der Konstruktionsabstand kann sich während eines Feature-Generator-Laufes verändern. Damit ist es möglich, mehrmals Anfragen an die Case Base zu stellen, da verschiedene Cases zurückgeliefert werden können. Die andere Möglichkeit, den samplingbasierten Merkmalsvergleich gewinnbringend in die Abstandsbestimmung zwischen Cases aufzunehmen, besteht darin, ihn als zweite Stufe bei der Auswahl der in

Frage kommenden ähnlichen Lernprobleme zu verwenden. In der ersten Phase einer Anfrage sucht die Case Base die k Cases heraus, die dem aktuellen Lernproblem aufgrund der Basisattributgewichtung am ähnlichsten sind. Dieser Schritt ist, was die Laufzeit anbelangt, sehr günstig. Es müssen für den aktuellen Fall einmal die Basisgewichte berechnet werden, danach können die eigentlichen Vergleiche in $O(m \cdot n)$ (mit m Anzahl der Cases in der Case Base und n Anzahl der Basisattribute) durchgeführt werden. In dieser Phase wird also eine Vorauswahl getroffen. Aus der Menge der k ähnlichsten Cases wird dann im zweiten Schritt mit Hilfe des samplingbasierten Merkmalsvergleiches eine weitere Auswahl von k' Fällen getroffen, deren konstruierte Features dann als Lösungsvorschläge zurückgeliefert werden. Da diese Vergleiche versprechen zwar eine Präzisierung der Fallähnlichkeit, sie beinhalten aber gleichzeitig einen höheren Laufzeitaufwand. Falls die Anzahl der atomaren Operationen in den konstruierten Features durch eine Konstante begrenzt ist, so sind pro Vergleich zwischen zwei Fällen $O(|X_i| \cdot |X_j| \cdot s)$ Berechnungen nötig. Dabei sind X_i, X_j die Mengen der konstruierten Features für t_i bzw. t_j, s ist die Anzahl der Stichproben.

6. Experimente

Dieses Kapitel beschäftigt sich mit der experimentellen Evaluierung des Case Base Ansatzes. In diesem Rahmen werden auch verschiedene Möglichkeiten, die Ergebnisse einer Case Base-Anfrage in einem Lernverfahren gewinnbringend einzusetzen, vorgestellt. Des Weiteren werden Anwendungsszenarien für den Einsatz einer Case Base in verschiedenen Lernverfahren entworfen und erläutert.

6.1. Allgemeine Vorraussetzungen

Zunächst werden im Folgenden die Vorraussetzungen der verschiedenen Experimente aufgeführt.

6.1.1. Programmumgebung

Die Experimente wurden mit dem Data Mining-Werkzeug RapidMiner (vorher Yale) [Mierswa et al. 2006] durchgeführt. Für diese Arbeit wurden insbesondere Feature Generatoren wie der "Yagga2"-Operator (siehe [Ritthoff et al. 2002]) sowie die Operatoren LinearRegression und W-PaceRegression [Wang und Witten 1999] des WEKA-Projekts [Witten und Frank 2005] eingesetzt.

6.1.2. Verwendete Case Base

Für den Aufbau der Case Base musste zunächst eine große Menge an Problemfällen bzw. Datensätzen bereitgestellt werden. Anscheinend existieren kaum reale Datensätz in entsprechender Anzahl, die einerseits ähnlich genug sind, dass sie ähnlich geartete Lösungen haben, andererseits aber nicht genau denselben funktionalen Zusammenhang zwischen Basisattributen und Ergebnis - also dieselbe Lösung - besitzen. Daher wurde ein synthetischer Ansatz bevorzugt. Dabei wurde für jeden Problemfall ein Funktionsbaum zufällig erzeugt, dessen innere Knoten arithmetische Operatoren wie z.B. binäre Summe, binäres Produkt, Quadratwurzel und Sinus sind, während die Blätter entweder Basisattribute oder Konstanten beinhalten (siehe Abbildung 6.1).

Der Wurzelknoten der Funktionsbäume ist ein n-ärer Summenoperator. Diese feste Belegung wurde gewählt, um dem linearen Lerner entgegenzukommen, der eine gewichtete Summe seiner Eingabeattribute bildet (siehe Abschnitt 2.3 und Abbildung 2.1). Weiterhin wurde bei der Erzeugung der Bäume darauf geachtet, die einzelnen Teilbäume unter der Wurzel etwa gleich groß werden zu lassen, gemessen an der Anzahl der enthaltenen Operatorknoten. Um mit diesen Funktionsbäumen schließlich Datensätze zu generieren, wurden für die im jeweiligen Baum benutzten Basisattribute Zufallszahlen erzeugt und in die durch den Funktionsbaum dargestellte Funktion eingesetzt, um den Zielwert zu berechnen. Die Gesamtzahl der in einem Funktionsbaum verwendeten Operatoren kann

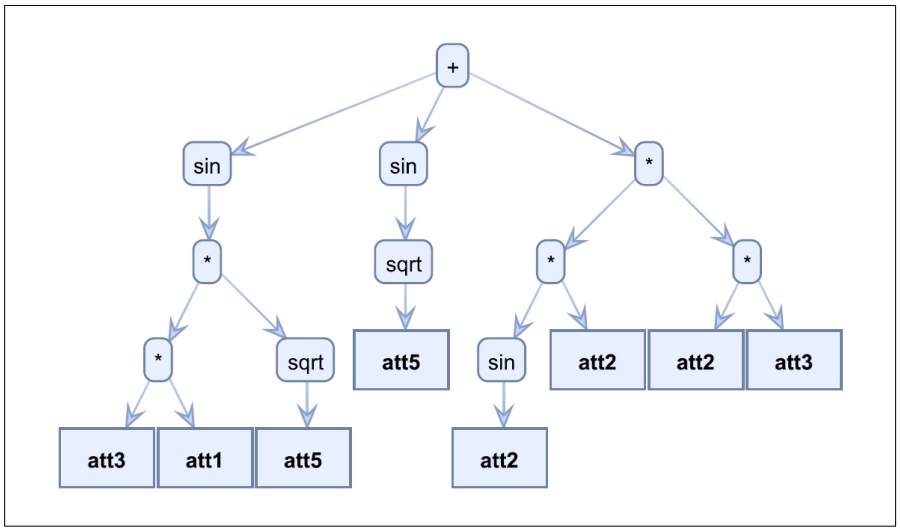


Abbildung 6.1.: Zufällig erzeugter Funktionsbaum mit 10 Operatoren (den fest vorgegebenen Summenoperator in der Wurzel nicht mitgezählt)

variiert werden, um unterschiedlich komplexe Zielfunktionen zu generieren, was sich wiederum auf die Schwierigkeit der Regression auf dem zugehörigen Datensatz auswirkt. Obwohl die Schwierigkeit einer Regressionsaufgabe auf diese Weise kaum formal definiert werden kann, zeigen die Ergebnisse der Experimente dennoch, dass sich in der Praxis auf diese Weise Testfälle mit unterschiedlichen Schwierigkeitsgraden für die betrachteten Lernverfahren erzeugen lassen.

Die so erzeugten Problemfälle können nun auf verschiedene Arten in (gelöste) Cases überführt werden, aus denen die Fallbasis aufgebaut wird. Ein solcher Fall besteht, wie in Abschnitt 2.5 erläutert, aus der Gewichtung der Basisattribute sowie den konstruierten Features in Form von Funktions(teil)bäumen. Zum einen können nun, wie in der Arbeit von [Mierswa und Wurst 2005b], die einzelnen Einträge durch Feature Generator-Läufe auf Datensätzen erzeugt werden. Dabei werden dann die vom Feature Generator erzeugten Featurekonstruktionen als Lösung für den jeweiligen Fall benutzt. Im realen Einsatz stellt diese Methode sicherlich die einzige Möglichkeit dar, Einträge zum Aufbau der Case Base zu gewinnen. Da bei den vom Autor durchgeführten Experimenten jedoch ohnehin synthetische Datensätze mit bekannter Zielfunktion verwendet wurden, konnten die Einträge der Case Base direkt erzeugt werden. Dazu mussten lediglich die zufällig erzeugten Teilbäume, die unter dem Wurzelknoten des zum Datensatz gehörenden Funktionsbaumes hängen, als konstruierte Features in den Case eingetragen werden. Diese Methode hat zwei Vorteile: Zum einen ist der Rechenaufwand zum Erzeugen eines Cases viel geringer, da der gesamte Feature Generator-Lauf unnötig ist, zum anderen ist die so erzeugte Lösung perfekt. Wenn ein Case Base-unterstützter linearer Lerner für einen Datensatz genau die konstruierten Attribute aus der Case Base erhält, mit denen er erzeugt worden ist, sollte der lineare Lerner daraus wieder die ursprüngliche Funktion zusammensetzen können und damit eine fehlerfreie Regression durchführen

können. Für die Featurekonstruktionen aus einem Generatorlauf ist nicht jedoch nicht garantiert, dass sie genauso gut zu dem Problemfall passen, für den sie erzeugt wurden.

6.1.3. Performanzmaß und Referenzexperimente

Die für die im Folgenden beschriebenen Experimente benutzten Datensätze wurden auf die gleiche Art erzeugt, wie im vorherigen Abschnitt beschrieben. Sowohl für den Aufbau der Case Base als auch für die Experimente wurden ausschließlich Datensätze mit 5 Basisattributen und 500 Instanzen verwendet. Um Referenzergebnisse zum Performanzvergleich zu erhalten, wurde für alle Datensätze zunächst eine einfache lineare Regression ohne Hinzufügung von konstruierten Attributen durchgeführt und die Performanz der resultierenden linearen Modelle anhand des sogenannten Root Relative Squared Error gemessen, der sich nach der Formel

$$F_{rrse} = \sqrt{\frac{\sum_{i=0}^{n}(Y_i - Pred_i)^2}{\sum_{i=0}^{n}(Y_i - \overline{Y_i})^2}}$$

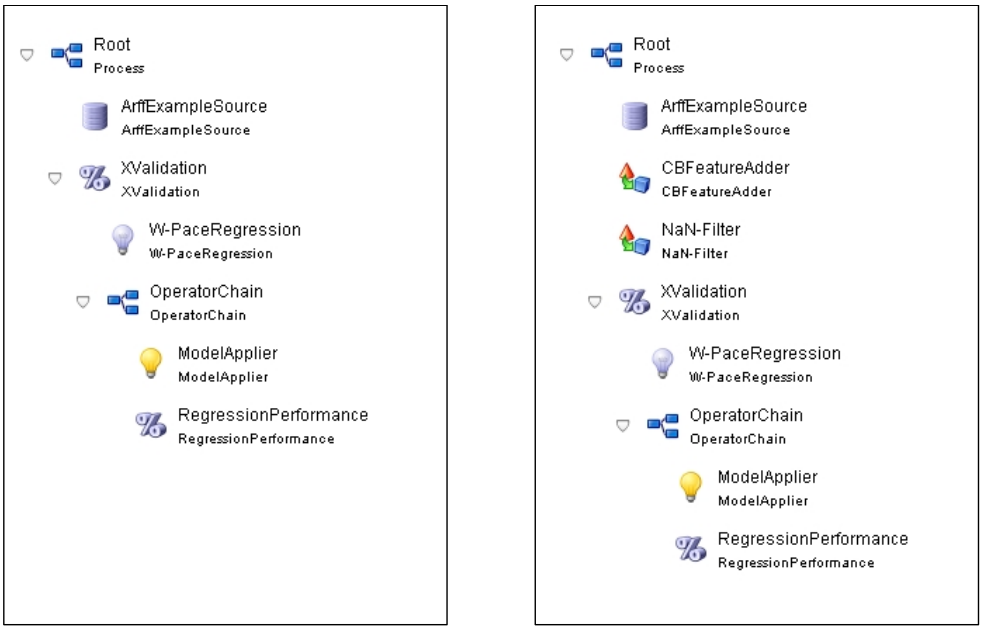
berechnet. Es wurde an dieser Stelle Wert darauf gelegt, ein relatives Fehlermaß zu benutzen, da alle Experimente mit mehreren Datensätzen wiederholt wurden, deren Zielwerte in unterschiedlichen Wertebereichen W liegen konnten. Ein absolutes Fehlermaß hätte dazu geführt, dass der Fehlerwert zu einem Datensatz mit großem W die Fehlerwerte zu Datensätzen mit kleinerem W ungerechtfertigterweise dominiert. Der Performanzwert dieser linearen Modelle stellt also sozusagen eine obere Grenze für die Performanz der Case Base-unterstützten Experimente dar, falls diese gegenüber der linearen Regression tatsächlich eine Verbesserung darstellen sollen.

Der Aufbau des Referenzexperimentes wird in Abbildung 6.2(a) dargestellt. Nachdem der jeweilige Testdatensatz geladen worden ist, wird in einer Kreuzvalidierungsschleife in jeder Iteration ein Teil der Instanzen des Datensatzes an den PaceRegression-Operator zum Erlernen eines linearen Modells übergeben, bevor anhand der verbliebenen Instanzen die Performanz des linearen Modells mittels des oben beschriebenen Fehlermaßes bestimmt wird.

6.2. Einmalige Erweiterung des Merkmalsraumes

Die erste Reihe von Experimenten befasst sich mit einer einfachen Erweiterung des Referenzexperimentes. Nach dem Einlesen des Trainingsdatensatzes werden die Basisattribute durch eine einmalige Anfrage an die Case Base um die konstruierten Features einer festgelegten Anzahl der ähnlichsten Cases ergänzt. Der so erweiterte Datensatz wird dann an einen linearen Lerner weitergegeben (siehe Abbildung 6.2(b)). Diese Prozedur wurde mit 10 Trainingsdatensätzen unter Benutzung der zehnfachen Kreuzvalidierung durchgeführt.

Bei diesen Experimenten ergab sich eine Reihe von Parametern und Variationsmöglichkeiten, von denen die Wichtigsten jeweils in einer eigenen Testreihe betrachtet wurden. Insbesondere waren dies die Anzahl der abgefragten Cases, die Größe der Case Base sowie das verwendete Distanzmaß.



(a) Referenzexperiment linearer Lerner (b) Einmalige Case Base-Anfrage

Abbildung 6.2.: Experimentaufbauten

6.2.1. Anzahl benutzter Cases (Experimentreihe 1a)

Die erste Testreihe beschäftigte sich mit den Auswirkungen, die verschieden große Mengen an Featurekonstruktionen aus den zurückgelieferten Cases auf die Regressionsperformanz des linearen Lerners haben. Es wurden dazu Testdaten verwendet, deren Funktionsbäume 50 Operatoren enthalten, das entspricht der Funktionskomplexität der für den Case Base-Aufbau benutzten Testdatensätze. Die benutzte Case Base beinhaltet 2000 Cases, die im Durchschnitt jeweils etwa 10 einzelne Featurekonstruktionen aufweisen. Neben dem bereits beschriebenen Referenzexperiment wurden in dieser Reihe Experimente durchgeführt, die 1, 10 und 40 Cases verwendeten, um den Eingaberaum des linearen Lerners zu erweitern. Diese Cases wurden anhand der Manhattandistanz und der SVM-Gewichtsvektoren aus der Case Base ausgesucht, wie in den Kapiteln 3 und 4 beschrieben. Als Lerner wurde in diesen Experimenten der Pace Regression-Algorithmus benutzt, da er laut [Wang und Witten 1999] in der Lage ist, mit irrelevanten Attributen umzugehen. Diese Fähigkeit ist besonders bei Benutzung von vielen Cases von Vorteil, denn sonst besteht die Gefahr, dass zu viele irrelevante Case Base Features die Performanz verschlechtern. Leider erhöht sich ab ca. 300 Attributen die Laufzeit des PaceRegression-Operators erheblich, so dass die teilweise deutlichen Verbesserungen der Performanz im Vergleich zum Referenzexperiment mit ebenso deutlich gestiegenen Laufzeiten der Experimente erkauft wurden. Tabelle 6.1 zeigt den durchschnittlichen relativen Fehler (RRSE) der einzelnen Experimente über 10 Datensätze. Die Spalte "Lin. Regression" enthält die Werte des beschriebenen Referenzexperiments, die drei weiteren Spalten jeweils die Ergebnisse bei Hinzunahme der konstruierten Attribute von 1, 10 bzw. 40 Cases. Die jeweils angegebene Laufzeit ist die Gesamtzeit, die die Experimente für alle

Performanz	Lin. Regression	1 Case	10 Cases	40 Cases
absolut (∅/Std.-Abw)	0,7390/0,43	0,6793/0,4	0,3205/0,27	0,0846/0,09
relativ ($\frac{absolut}{Lin-Reg}$)	1	0,922	0,353	0,33
Signifikanz (t-Wert/α)		1,969/7,7%	3,892/0,3%	5,045/0,05%
Laufzeit [sec.]	3	29	51	1611

Tabelle 6.1.: Testreihe 1a: Referenzexperimentreihe und Experimentreihen mit konstruierten Features aus 1, 10 und 40 Case Base-Fällen (ausgesucht mittels Manhattendistanz), jeweils über 10 Datensätze. Zeile "absolut" enthält Durchschnitt und Std.-abweichung der Regressionsfehler (rrse), Zeile "relativ" den auf den Referenzwert normierten Durchschnittswert, Zeile "Signifikanz" die Ergebnisse der paarweisen t-Tests, jeweils auf die lin. Reg. bezogen. Die Laufzeiten sind jeweils aufsummiert über alle 10 Durchläufe.

10 Datensätze benötigt haben.

Auffällig an diesen Ergebnissen ist, dass mit steigender Anzahl an Cases eine signifikante Verbesserung der Performanz gegenüber dem nicht Case Base-unterstützten linearen Lerner sichtbar wird. Die paarweisen t-Tests, mit denen die Signifikanz der Unterschiede in den Ergebnissen geprüft wurde, haben ergeben, dass lediglich unter Hinzufügung von Featurekonstruktionen aus nur einem Case das übliche Signifikanzniveau von $\alpha = 5\%$ knapp verfehlt wird. Bei den Experimenten mit Hinzufügung von 10 bzw. 40 Cases wird es dagegen deutlich unterschritten. Dies könnte auf die relativ hohe Komplexität der für die Erzeugung der Testdatensätze verwendeten Funktionsbäume zurückzuführen sein.

6.2.2. Distanzmaße (Experimentreihe 1b)

Die nächste Testreihe beschäftigt sich mit dem Einfluss verschiedener Distanzmaße auf die Performanz. Neben der bereits in Experimentreihe 1a verwendeten Manhattandistanz wurden hier die euklidische sowie die Overlap-Distanz betrachtet und schließlich noch eine Experimentreihe mit zufälliger Case-Auswahl durchgeführt. Für jede dieser drei neuen Distanzen wurden die Experimente mit 1, 10 und 40 Cases aus Experimentreihe 1a wiederholt. Während sich der Trend aus der vorherigen Experimentreihe bzgl. der Anzahl der verwendeten Cases hier auch wieder zeigt, scheint die Wahl des Abstandsmaßes zur Auswahl der Cases kaum einen nennenswerten Einfluß auf die Performanz zu haben. Dieser Eindruck kann mit Hilfe der Varianzanalyse (ANOVA, ANalysis Of VAriances) untermauert werden: Dabei wird das verwendete Distanzmaß als Faktor angesehen, so dass sich folgende Wahrscheinlichkeiten für die Hypothese "Die Wahl des Distanzmaßes beeinflusst die Regressionsperformanz nicht" ergeben: Im Fall mit jeweils einem hinzugefügten Case 99,7% (F = 0,017), für die Experimente mit 10 Cases 99,2% (F = 0,034) und für 40 Cases 95,9% (F = 0,101). Trotz dieser scheinbar recht starken Hinweise auf eine Indifferenz des Case Base-Ansatzes bezüglich des verwendeten Distanzmaßes bleibt Folgendes zu bemerken: Die schlechte Bewertung der oben genannten Hypothese entsteht vor allem aufgrund der großen Standardabweichung bzw. Varianz der Performanzwerte für die einzelnen Datensätze. Diese Varianzen ergeben sich jedoch nicht aus dem Case

	Manhattan	Euklid	Overlap	Zufällig
1 Case	0,6793/0,4	0,6781/0,41	0,7304/0,44	0,7183/0,43
10 Cases	0,3205/0,27	0,3137/0,26	0,3791/0,28	0,3591/0,35
40 Cases	0,0846/0,09	0,0853/0,09	0,0741/0,08	0,1434/0,13

Tabelle 6.2.: Testreihe 1b: Experimentreihen mit verschieden großen Mengen an Featurekonstruktionen aus der Case Base (vertikal) und unterschiedlichen Distanzmaßen zur Auswahl der Cases (horizontal). Jeweils Mittelwert und Standardabweichung von Experimenten an 10 Datensätzen

Base-Ansatz und den verwendeten Distanzmaßen, sondern vielmehr aus der Struktur der Testdatensätze. Dies ist daran zu erkennen, dass bereits die Ergebnisse der einfachen linearen Regression ohne Case Base-Unterstützung eine vergleichbar hohe Standardabweichung aufweisen (siehe Tabelle 6.1). Wenn nun bei der Analyse der Ergebnisse der Fokus etwas auf die Mittelwerte verschoben wird, so ergeben sich dennoch Hinweise, dass es immerhin einen kleinen Unterschied macht, welches Distanzmaß in einer Experimentreihe Verwendung findet: Zwar ist der Unterschied zwischen der Manhattan- und der Euklidischen Distanz so gering, dass er vernachlässigt werden kann, die Overlap-Distanz fällt hingegen schon etwas deutlicher ab, sogar unter die Performanz der Experimente mit zufälliger Case-Auswahl (siehe auch Abbildung 6.3). Insgesamt gesehen liegen die mit den verschiedenen Distanzen erzielten Performanzwerte jedoch recht nahe beieinander, ähnlich wie schon bei [Mierswa und Wurst 2005b].

6.2.3. Umfang der Case Base (Experimentreihe 1c)

Eine weitere interessante Frage ist die nach dem Einfluss des Umfangs der Case Base auf die Ergebnisse der Experimente. Sicherlich ist bei einer höheren Anzahl von Cases die Wahrscheinlichkeit, ein oder mehrere passende Einträge für das aktuelle Problem zu finden, größer als bei einer kleinen Case Base. Andererseits ist aber kaum darauf zu hoffen, alle möglichen Funktionen in einer Case Base abbilden zu können, selbst unter den für diese Experimente getroffenen Einschränkungen auf einige wenige Basisoperatoren, aus denen sich die betrachteten Funktionen zusammensetzen. Des Weiteren sind der Größe einer Case Base Grenzen gesetzt durch zwei Umstände: Zum einen ist die Erzeugung einer Case Base zeitaufwändig, vor allem dann, wenn die konstruierten Features aus einem Generatorlauf gewonnen werden müssen. Zum anderen wird der Platzbedarf und Verwaltungsaufwand für eine zu große Case Base zum Problem. Ab einer bestimmten Menge an Cases dürfte auch der Vergleich der Attributgewichte zu einer so starken Erhöhung der Laufzeit eines Experimentes führen, dass die Benutzung dieser Case Base als unwirtschaftlich angesehen werden muss. Daher macht es an dieser Stelle durchaus Sinn zu untersuchen, ob eine Vergrößerung der Case Base im Durchschnitt überhaupt zu signifikanten Performanzverbesserungen führt. Neben der schon in den vorangegangenen Experimentreihen benutzten Case Base mit 2000 Einträgen wurden hier eine kleinere Case Base mit 500 Cases und eine größere mit 10000 Cases zum Vergleich herangezogen. In Tabelle 6.3 und Abbildung 6.4 werden die 3 Case Bases mit CB_{500}, CB_{2000} und CB_{10000}

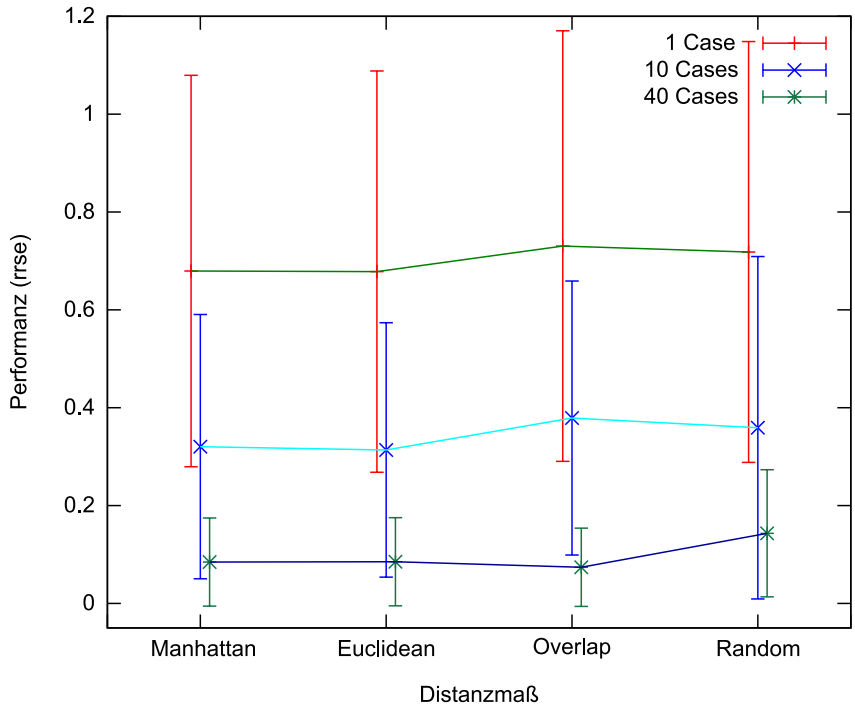


Abbildung 6.3.: Testreihe 1b: Durschnittsperformanzen und Standardabweichungen bei verschiedenen Anfragegrößen und Distanzmaßen. Die leichten Verschiebungen der Kurven auf der X-Achse dienen lediglich der Lesbarkeit

bezeichnet. Die Datensätze für alle drei Case Bases wurden mit den gleichen Einstellungen des Funktionsbaumgenerators erzeugt, nämlich mit den Basisoperatoren Summe, Produkt, Quadratwurzel und Sinus sowie der Vorgabe, dass jeder Funktionsbaum aus 50 dieser Operatoren zusammengesetzt ist. Außerdem gilt $CB_{500} \subset CB_{2000} \subset CB_{10000}$, d.h. die Fälle der kleineren Case Base sind jeweils in der Menge der Fälle der größeren Case Base enthalten.

Die Ergebnisse deuten darauf hin, dass die Größe der Case Base zumindest bei Anfragen mit wenigen Cases kaum einen Einfluß zu haben scheint. Die Varianzanalysen ergeben folgende Wahrscheinlichkeiten für die Signifikanz des Einflusses der Case Base-Größe:

- Versuchsreihe mit einem hinzugefügten Case: 1,0% (F = 0,01)
- Versuchsreihe mit 10 hinzugefügten Cases: 0,7% (F = 0,007)
- Versuchsreihe mit 40 hinzugefügten Cases: 4,4% (F = 0,045)

	CB_{500}	CB_{2000}	CB_{10000}
1 Case	0,6801/0,42	0,6793/0,4	0,6454/0,38
10 Cases	0,2960/0,23	0,3205/0,27	0,314563/0,22
40 Cases	0,1089/0,12	0,0846/0,09	0,0688/0,06

Tabelle 6.3.: Testreihe 1c: Experimentreihen mit verschieden großen Mengen an Featurekonstruktionen aus der Case Base (vertikal) und unterschiedlichen Case Base-Größen (horizontal). Jeweils Mittelwert und Standardabweichung von Experimenten an 10 Datensätzen

6.3. Case Base-Unterstützung eines evolutionären Feature Generators

Während in den vorangegangenen Abschnitten lediglich eine einmalige Erweiterung des Attributraumes durch nichtlineare Features stattgefunden hat, wird in den folgenden Experimenten wesentlich mehr Zeit darauf verwendet, einen optimalen Raum von Eingabeattributen zu finden, mit denen ein linearer Lerner ein nichtlineares Problem lösen kann (siehe Abschnitte 2.3 und 2.4). Eine Möglichkeit dazu ist die Benutzung eines evolutionären Feature Generators. Ein solcher Algorithmus wird in [Ritthoff et al. 2002] beschrieben und ist in der Lernumgebung RapidMiner im Operator Yagga2 implementiert. Da dieser Operator ein Meta-Lernverfahren durchführt, ändert sich der Experimentaufbau im Vergleich zu den vorangegangene Abschnitten geringfügig: Wie in Abbildung 6.5 zu sehen ist, wird die gesamte Kreuzvalidierungsschleife in den Operator Yagga2 eingebettet. Dies spiegelt die Tatsache wider, dass die evolutionäre Feature Generation den inneren linearen Lerner und die Performanzmessung zur wiederholten Evaluierung der erzeugten Attributräume benutzt.

Für die folgenden Experimente wurde vom Autor der Operator Yagga2 um eine Abfragemöglichkeit für die Case Base erweitert. Der so entstandene Operator Yagga3 benutzt neben den klassischen Evolutionsoperatoren wie Mutation, Crossover und Attributverknüpfung zusätzlich die Case Base, um an ein Individuum weitere konstruierte Attribute aus gefundenen Cases anzufügen. Aufgrund der Ergebnisse der vorangegangenen Experimente, bei denen eine deutliche Performanzsteigerung durch die Benutzung von weit mehr als einem Case beobachtet wurde, erschien es sinnvoll, auch den Experimenten mit dem Feature Generator mehrere Cases auf einmal abzufragen. Soweit nicht im jeweiligen Abschnitt ausdrücklich anders beschrieben, wurden pro Anfrage zunächst mit Hilfe der Manhattan-Distanz und den Attributgewichtsvektoren 15 ähnliche Fälle herausgesucht, aus denen dann per samplingbasiertem Merkmalsvergleich (siehe 5.1.2) schließlich 5 Cases ausgewählt wurden, deren konstruierte Attribute dem aktuellen Individuum hinzugefügt wurden. Die Häufigkeit, mit welcher für ein Individuum eine solche Case Base-Anfrage gestellt wurde, läßt sich im Operator per Wahrscheinlichkeitswert einstellen, als Voreinstellung wurde ein Wert von 0,5 gewählt. Dies erscheint zunächst sehr hoch angesetzt, jedoch relativiert sich dieser Wert wieder durch den generationsbasierten Filter, dem der Operator der Case Base-Anfrage unterliegt: Per Parameter kann eingestellt werden, dass der Operator nur alle n Generationen angewendet wird. Für die folgenden Experimente

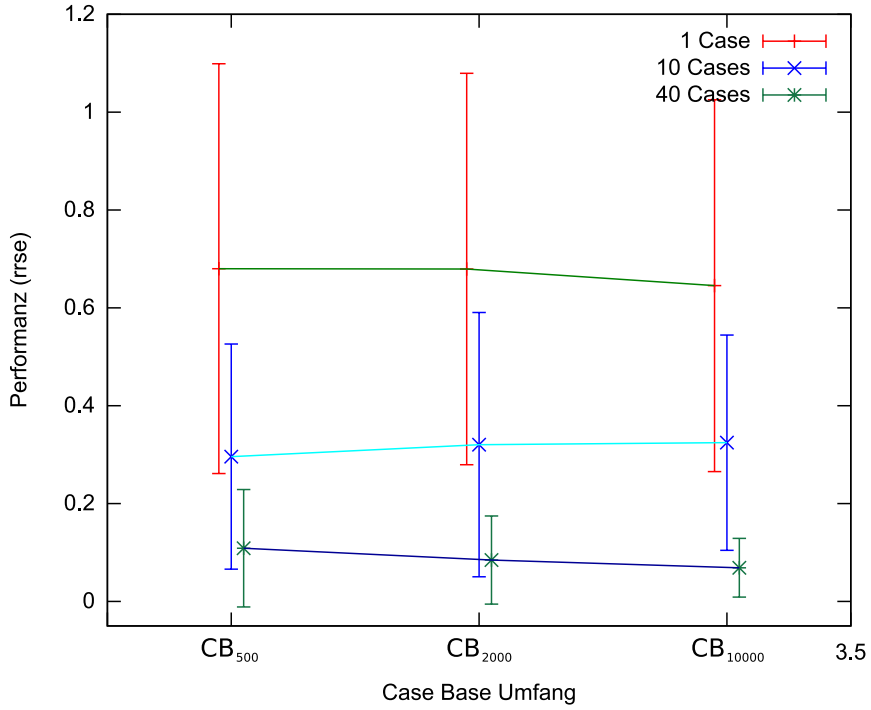


Abbildung 6.4.: Testreihe 1c: Durschnittsperformanzen und Standardabweichungen bei verschiedenen Anfrage- und Case Base-Größen. Die leichten Verschiebungen der Kurven auf der X-Achse dienen lediglich der Lesbarkeit

wurde $n = 5$ gesetzt.

Um auch bei den folgenden Experimentreihen eine faire Bewertung des Case Base-Ansatzes vornehmen zu können, wurde zunächst wieder eine Referenzexperimentreihe ohne Verwendung der Case Base unter Benutzung des oben vorgestellten RapidMiner-Operators Yagga2 durchgeführt. Deren Ergebnisse werden jeweils als Vergleichswerte bei den Experimentreihen, die die Case Base verwendeten, mit angegeben.

Die ersten beiden Experimentreihen widmen sich dem direkten Vergleich zwischen klassischer evolutionärer Feature Generation und der Case Base-unterstützten evolutionären Merkmalskonstruktion. Es ist nicht zu erwarten, dass die Case Base-Unterstützung eine bessere Performanz als das klassische Verfahren liefert, wenn keine zeitliche Begrenzung gesetzt wird. Der Grund dafür ist, dass der Feature Generator jede beliebige Featurekonstruktion, die sich aus seinen Basisoperatoren überhaupt zusammensetzten läßt (also auch jede, die in der Case Base abgelegt sein kann), mit einer Wahrscheinlichkeit $p > 0$ irgendwann auch tatsächlich baut (siehe [Ritthoff et al. 2002], Kapitel 2). Der Vorteil, den die Benutzung der Case Base einbringt, kann also nur darin liegen, dass das herkömmliche Verfahren durch sie abgekürzt bzw. beschleunigt wird. Um zu bestimmen, ob die Case Base-Unterstützung tatsächlich einen solchen Vorteil einbringt, gibt es zwei Möglichkeiten: Zum einen kann man einen Zielperformanzwert vorgeben und vergleichen, nach welcher Generationenzahl oder Rechenzeit die beiden Verfahren diesen Zielwert er-

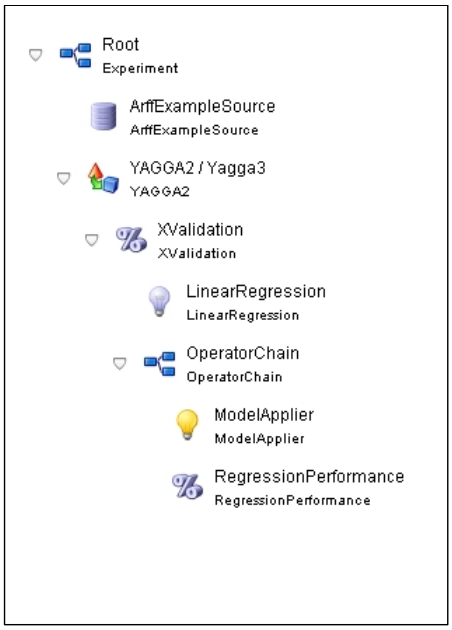


Abbildung 6.5.: Experimentaufbau Feature Generator

reichen. Die zweite Möglichkeit besteht darin, einen zeitlichen Grenzwert vorzugeben und dann die erreichten Performanzwerte der beiden Verfahren zu vergleichen. Leider ist die erste Methode für Experimentreihen mit mehren Datensätzen kaum praktikabel, da die Zielperformanzwerte, welche beide Verfahren innerhalb akzeptabler Zeiträume erreichen können, von Datensatz zu Datensatz stark schwanken. Aus diesem Grund verwenden die beiden ersten Experimentreihen jeweils eine Art der Zeitbeschränkung, um die Feature Generatorläufe zu beenden.

6.3.1. Beschränkung der Generationsanzahl (Experimentreihe 2a)

Die erste Reihe von Experimenten vergleicht die beiden konkurierenden Verfahren anhand der durchschnittlichen Performanz, die sie nach einer festgelegten Anzahl von Generationen erreichen. Dazu wurden jeweils 10 Trainigsdatensätze in einer Schleife nacheinander bearbeitet mit einer festgelegten Anzahl an Generationen und Individuen, die jeweils von einer Generation in die nächste übernommen wurden.

Tabelle 6.4 und Abbildung 6.6 stellen die Ergebnisse der Testreihe dar, die mit den gleichen Datensätzen durchgeführt wurde, die schon in Abschnitt 6.2 verwendet wurden. Aufgeführt sind die Ergebnisse für die einzelnen Datensätze nach jeweils 20 Generationen, darunter die durschnittliche Performanz und Standardabweichung (gemessen mit dem Root Relative Squared Error) sowie die Laufzeit für alle Datensätze. Zu sehen sind größtenteils deutliche Verbesserungen der Ergebnisse durch den Einsatz der Case Base, die jedoch leider mit einer drastisch erhöhten Laufzeit einhergehen, trotz der gleichen Generationenzahl in allen Experimentreihen. Dies liegt vor allem darin begründet, dass die Individuen, die der Yagga3 erzeugt im Durchschnitt wesentlich mehr Features als die

Nr.	Yagga2	Yagga3 (2 Cases)	Yagga3 (5 Cases)
1	0,1745	0,163	0,068
2	0,8499	0,4687	$1,955 * 10^{-15}$
3	0,0475	0,0372	0,0265
4	0,9971	0,359	0,0989
5	0,0957	0,0356	0,0284
6	0,9675	0,3254	0,1491
7	0,7373	0,5711	0,0206
8	0,832	0,4478	0,0595
9	0,997	0,6702	0,2642
10	0,9741	0,7045	0,3538
∅/Std.Abw.	0,6673/0,4	0,3783/0,24	0,1069/0,11
Laufzeit	638 sec.	2021 sec.	13337 sec.

Tabelle 6.4.: Testreihe 2a: Beschränkung auf 20 Generationen (10 Individuen), Performanzen von Yagga2, Yagga3 mit 2 und 5 Cases pro Anfrage, Laufzeiten jeweils aufsummiert über alle 10 Experimente

Individuen des Yagga2 beinhalten, was die Laufzeit des linearen Lerners deutlich erhöht. Aus diesem Grund wurden zwei Durchläufe mit dem Yagga3-Operator durchgeführt. Der erste Lauf (in Tabelle 6.4 mit "Yagga3 (2 Cases)" überschrieben) stellt einen Kompromiss zwischen Laufzeit und Regressionsperformanz dar, da bei diesem anstatt der sonst verwendeten 5 Cases pro Case Base-Anfrage nur 2 benutzt wurden. Durch diese Reduktion veringerte sich die durchschnittliche Anzahl der Attribute soweit, dass die Gesamtlaufzeit gegenüber der Experimentreihe mit 5 Cases wieder deutlich abnahm.

Bei der Betrachtung der Einzelergebnisse fällt insbesondere das Lernproblem Nr.2 auf, welches vom Yagga3 mit 5 Cases beinahe fehlerfrei gelöst werden konnte, aber auch bei den Fällen Nr.4, 7 und 8 zeigte der Yagga3 in beiden Ausführungen deutliche bessere Ergebnisse als der herkömmliche Yagga2. Zur Überprüfung der Signifikanz wurden mehrere paarweise t-Tests durchgeführt. Zunächst wurden die Ergebnisse des Yagga2-Experimentes mit denen des Referenzexperimentes aus Abschnitt 6.2 verglichen (siehe etwa Tabelle 6.1, Spalte "Lin. Regression"). Dieser Test zeigte keine signifikante Verbesserung der Regressionsperformanz durch die Benutzung des Yagga2. Im Folgenden wurden dann gleichermaßen die Ergebnisse der Reihen "Yagga2" mit "Yagga3 (2 Cases)" und "Yagga3 (2 Cases)" mit "Yagga3 (5 Cases)" verglichen. Hierbei zeigten sich durchaus signifikante Verbesserungen (t-Wert 3,783, $\alpha = 0{,}43\%$ bzw. t-Wert 7,346, $\alpha < 0{,}01\%$).

6.3.2. Beschränkung der Laufzeit (Experimentreihen 2b und 2c)

Wie im vorangegangenen Abschnitt 6.3.1 gezeigt, lassen sich mit dem Case Base Ansatz durchaus bessere Performanzen in der gleichen Anzahl an Generationen erreichen. Da der Yagga3-Operator dabei jedoch aufgrund der größeren Menge an betrachteten konstruierten Attributen wesentlich mehr Rechenzeit benötigt als der Yagga2-Operator, ist dadurch noch nicht gezeigt, dass der Ansatz der Case Base-Unterstützung wirklich die

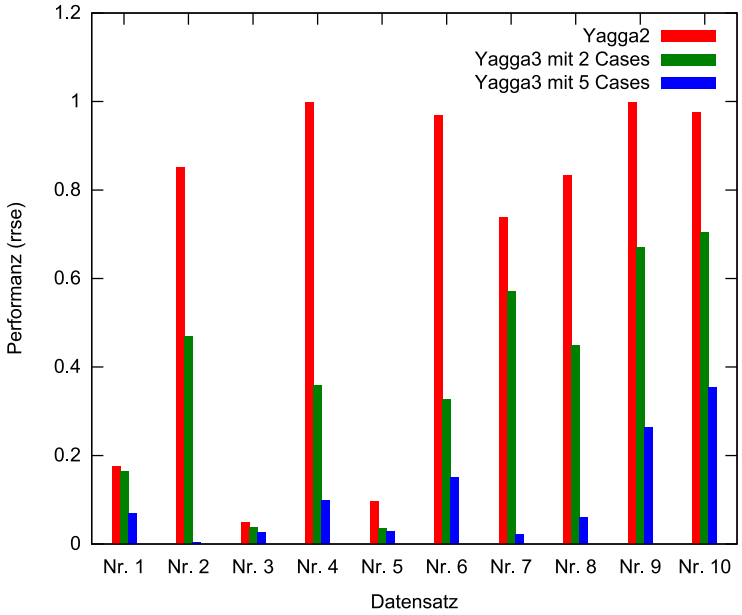


Abbildung 6.6.: Testreihe 2a: Beschränkung auf 20 Generationen (10 Individuen), Performanzen von Yagga2, Yagga3 mit 2 und 5 Cases pro Anfrage

Leistungsfähigkeit evolutionärer Feature Generation erhöht. Zu diesem Zweck sind vom Autor weitere Experimente durchgeführt worden. Dazu wurden beide Yagga-Operatoren mit einem neuen Abbruchkriterium ausgestatten. Die so modifizierten Operatoren brechen die Optimierung nach einer vorgegebenen Laufzeit ab und übergeben ihr bis dahin bestes Individuum [1].

In den beiden Experimentreihen 2b und 2c, deren Ergebnisse in den Tabellen 6.5 und 6.6 und den zugehörigen Abbildungen 6.7 und 6.8 dargestellt sind, wurde den Yagga-Operatoren pro Datensatz jeweils 100 bzw. 200 Sekunden Zeit gegeben, bevor der nächste Datensatz bearbeitet werden musste. Die so erzwungene vergleichbare Laufzeit von Yagga2 und Yagga3 ermöglicht es, eine faire Gegenüberstellung zwischen beiden Verfahren durchzuführen auf der Basis der in dieser Laufzeit erreichten Regressionsperformanz. Auch bei diesem Vergleich zeigt sich, dass die Case Base-Unterstützung einen Vorteil gegenüber der klassischen evolutionären Feature Generation einbringt: Während der Yagga2-Operator weder mit einer Laufzeit von 100 noch mit 200 Sekunden einen nennenswerten Vorteil gegenüber dem Ergebnis einer einfachen linearen Regression (∅-Performanz: 0,7390) einbringt, verringert sich der Fehler unter Benutzung der Case Base deutlich, im Durchschnitt etwa um die Hälfte. Wie oben bereits vermutet, liegt die Stärke des Case Base-Ansatzes also anscheinend in einer Beschleunigung der Konvergenz der Feature Generation. Dieser These geht der nächste Abschnitt (6.3.3) nach.

[1] Genauer: Es wird nach der Generation abgebrochen, in welcher die vorgegebene Laufzeit abläuft. Die Bearbeitung der letzten Generation wird also ggfs. noch beendet

Nr.	Yagga2 100 Sek.	Yagga3 100 Sek.
1	0,1436	0,1035
2	0,6827	0,6907
3	0,0442	0,035
4	0,982	0,2377
5	0,0955	0,0419
6	0,97	0,3598
7	0,9752	0,1504
8	0,9925	0,3902
9	0,9963	0,5582
10	0,9748	0,3566
∅/Std.Abw.	0,6857/0,42	0,2924/0,22

Tabelle 6.5.: Testreihe 2b: Yagga2 und Yagga3 mit jeweils 10 Individuen und 100 Sek. Laufzeit pro Datensatz. Einzelergebnisse und Mittelwert/Standardabweichung

Nr.	Yagga2 200 Sek.	Yagga3 200 Sek.
1	0,1427	0,0725
2	0,9871	0,0182
3	0,0436	0,0311
4	0,9825	0,1562
5	0,0954	0,0317
6	0,916	0,238
7	0,997	0,0491
8	0,8276	0,1926
9	0,9922	0,556
10	0,7289	0,3584
∅/Std.Abw.	0,6713/0,41	0,1704/0,17

Tabelle 6.6.: Testreihe 2c:Yagga2 und Yagga3 mit jeweils 10 Individuen und 200 Sek. Laufzeit pro Datensatz. Einzelergebnisse und Mittelwert/Standardabweichung

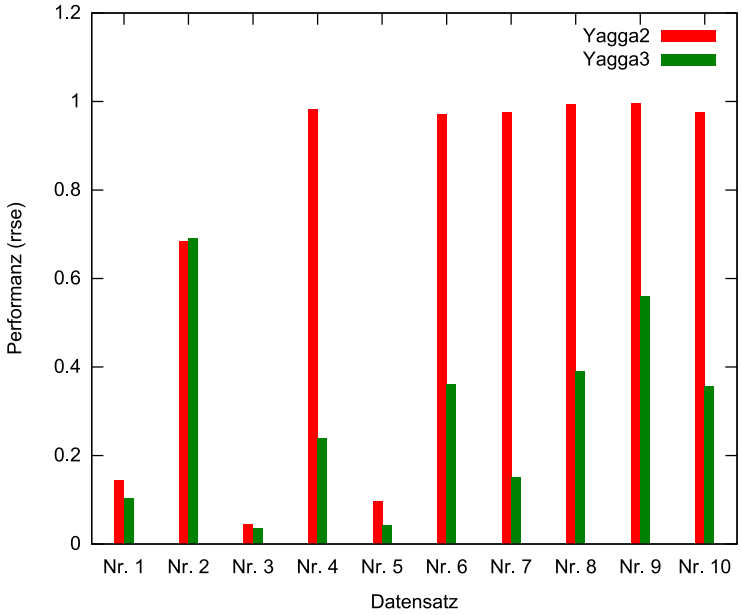


Abbildung 6.7.: Testreihe 2b: Beschränkung auf 100 Sek. Laufzeit pro Datensatz, Performanzen von Yagga2 und Yagga3

6.3.3. Konvergenzbetrachtung zur Feature Generation

In diesem Abschnitt soll der Effekt, den die Einbringung von Attributkonstruktionen aus der Case Base in den Feature Generator-Lauf hat, genauer untersucht werden. Zu diesem Zweck wurde ein einzelner Datensatz mit Hilfe der beiden RapidMiner-Operatoren Yagga2 und Yagga3 bearbeitet und der Verlauf des jeweils besten Performanzwertes über die Generationen betrachtet. Beide Läufe wurden über 30 Generationen durchgeführt, beim Durchlauf des Yagga3 wurden in der 5., 15. und 25. Generation Anfragen an die Case Base gestellt.

Es ist zu vermuten, dass sich eine Anfrage an die Case Base im Performanzverlauf niederschlägt, und zwar in Form einer sprungartigen Verbesserung der Performanz nach dem Einfügen der Case Base Features. Der klassische Feature Generator Lauf sollte dagegen einen gleichmäßigeren Verlauf der Performanz über der Zeit aufweisen. Einschränkend muss natürlich erwähnt werden, dass die Feature Generation schon im Allgemeinen aufgrund ihrer evolutionären Operatoren einen recht sprunghaften Verlauf auf der Performanzkurve erzeugt. Abbildung 6.9 zeigt jedoch den Unterschied zwischen beiden Methoden recht deutlich auf. Gut zu erkennen sind die Sprünge in der Verlaufskurve des Yagga3-Experimentes unmittelbar nach dem Einfügen der Case Base-Attribute in der 5. und 15. Generation. Die Veränderung nach der 25. Generation ist zwar nicht mehr so stark ausgeprägt wie an den beiden vorher genannten Punkten, trotzdem ist auch dort noch kleiner Performanzgewinn erkennbar.

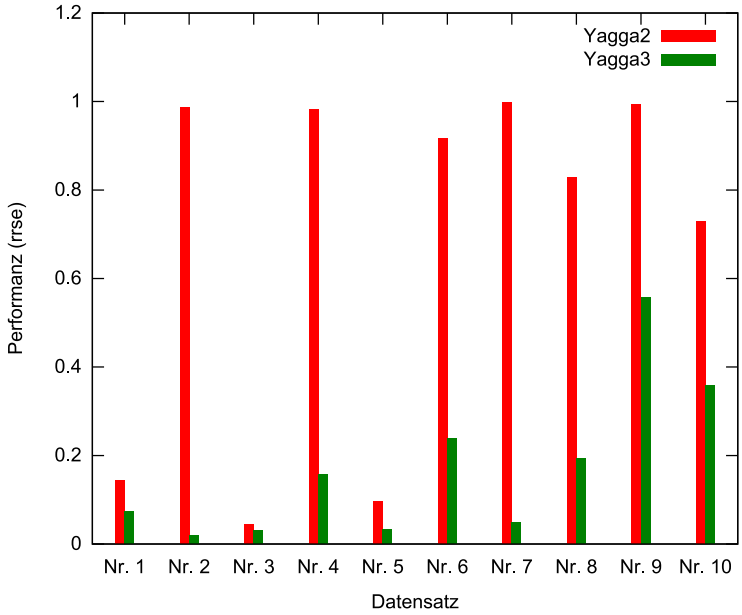


Abbildung 6.8.: Testreihe 2c: Beschränkung auf 200 Sek. Laufzeit pro Datensatz, Performanzen von Yagga2 und Yagga3

6.4. Zusammenfassung

Zum Ende dieses Experimentkapitels sollen zunächst noch einmal die wichtigsten hier getesteten Verfahren einander gegenübergestellt und direkt miteinander verglichen werden. Dazu wurde eine abschließende Experimentreihe mit 10 Datensätzen durchgeführt, in deren Verlauf ein einfacher linearer Lerner mit Basisattributen, ein linearer Lerner mit hinzugefügten Case Base-Attributen sowie die Feature Generatoren Yagga2 und Yagga3 Funktionsregressionen durchführten. Die Ergebnisse mit Angabe der insgesamt für alle 10 Datensätze jeweils benötigten Laufzeiten ist in Tabelle 6.7 angegeben.

Die Ergebnisse dieser letzten Experimentreihe und des gesamten Kapitels zeigen den positiven Effekt des Case Base Ansatzes relativ deutlich. Die Stärke dieses Effektes kann zwar von Fall zu Fall durchaus variieren, sie ist jedoch fast immer messbar, wie man sowohl an den Einzelergebnissen wie auch an den Durschnittswerten der Experimentreihen sieht. Leider zeigte sich aber vor allem in Abschnitt 6.2, dass der Einfluss von Parametern wie der Größe der Case Base oder dem Abstandsmaß in der praktischen Anwendung nicht so groß ist, wie während der theoretischen Untersuchung unterstellt wurde. Nichtsdestotrotz stellt die Case Base-Unterstützung eine echte Bereicherung bereits bestehender Methoden zur Regression dar, wie man sehr deutlich an den Ergebnissen der Experimentreihen 2b und 2c (Abschnitt 6.3.2) sieht. Dort wurden innerhalb vergleichbarer Laufzeiten durch die Benutzung einer Case Base signifikant bessere Fehlerraten erzielt.

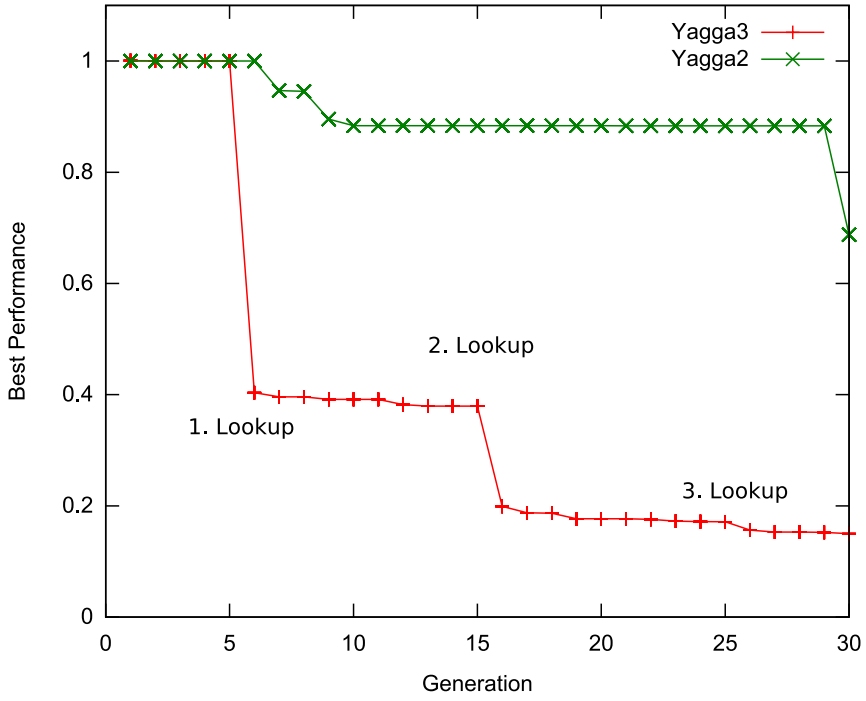


Abbildung 6.9.: Vergleich der Performanzentwicklung über die Generationen bei Yagga2 und Yagga3

	Linearer Lerner	erw. lin. Lerner	Yagga2	Yagga3
1	2,2299	0,1907	0,1865	0,1846
2	$5,53 * 10^{-12}$	$6,91 * 10^{-13}$	$6,81 * 10^{-13}$	$2,3 * 10^{-13}$
3	$3,93 * 10^{-12}$	$7 * 10^{-13}$	$6,93 * 10^{-13}$	$5,25 * 10^{-13}$
4	1,0107	0,026	0,029	0,0213
5	2,7997	0,4341	0,4268	0,3395
6	0,8485	1,0092	0,9448	0,5614
7	1,1786	0,0699	0,0692	0,0394
8	1,2994	0,0232	0,0227	0,0219
9	0,9855	0,0433	0,0426	0,0334
10	0,806	0,0478	0,0427	0,0233
∅/Std.Abw.	1,1158/0,87	0,1844/0,32	0,1764/0,3	0,1225/0,19
Laufzeit [sec.]	5	79	1380	3519

Tabelle 6.7.: Testreihe 3: Gesamtüberblick über alle verglichenen Verfahren an 10 Datensätzen

7. Matching von Basisattributen

Die bisher verfolgten Ansätze zum Vergleich zweier Lernaufgaben erfolgten unter der Voraussetzung, dass die Zuordnung der Basisattribute von Fall zu Fall immer diesselbe bleibt. Das bedeutet, dass jedes Lernproblem in der Case Base die gleichen Basisattribute haben muss mit jeweils der gleichen Bedeutung. Diese Voraussetzung scheint jedoch eher praxisfern zu sein, da eine solche Case Base zum einen unter Umständen nur schwierig mit genügend Fällen zu füllen ist, um hilfreich zu sein, zum anderen wäre sie aber auch so spezialisiert, dass ihr Einsatzgebiet nur sehr begrenzt wäre.

Daher besteht eine interessante Erweiterungsmöglichkeit des Case Base-Ansatzes darin, diese Voraussetzung aufzuheben. Zu diesem Zweck ist es nötig, vor der eigentlichen Suche nach einem ähnlichen Fall ein Attribut-Mapping herzustellen zwischen dem aktuellen Fall und allen in der Case Base abgelegten Fällen.

7.1. Problemstellung

Die Nutzbarmachung einer breiter gefächerten Menge an strukturell unterschiedlichen Datensätzen und ihrer Lösungen ist zwar ein lohnendes Ziel, sie ist aber mit mehreren Problemen verbunden. Das erste Problem ist das Fehlen jeglicher Zuordnung zwischen den Basisattributen zweier Datensätze. Selbst bei zwei Datensätzen, die denselben funktionalen Zusammenhang aufweisen, deren Attribute aber in unterschiedlicher Reihenfolge vorliegen, stößt das bisher beschriebene Verfahren an seine Grenzen. Als zweites Problem kommt hinzu, dass auch Datensätze mit unterschiedlicher Attributanzahl verglichen werden sollen. In diesem Fall gibt es gar keine eineindeutige Zuordnung zwischen den Attributen der beiden Fälle mehr.

Zur weiteren Betrachtung der Problematik soll diese zunächst formalisiert werden: Für eine neue Lernaufgabe t_{neu} mit den Basisattributen $X_{i=1..n}$ und eine bereits in der Case Base abgelegte Lernaufgabe t_{CB} mit den Basisattributen $Z_{j=1..m}$ soll eine Zuordnung (Matching) $\xi \in M$ gefunden werden, die jedem $X_i \in t_{neu}$ ein $Z_j \in t_{CB}$ zuordnet (siehe Abbildung 7.2). Weiterhin wird eine Bewertungsfunktion $\phi : M \rightarrow \mathbb{R}$ für diese Zuordnungen benötigt, um eine Aussage über die Qualität des Matchings treffen zu können. Mit Hilfe dieser Bewertungsfunktion ist es dann möglich, aus der Gesamtmenge aller Fälle der Case Base eine Vorauswahl zu treffen an Datensätzen, deren Attribute sich zumindest so gut zu den Attributen des aktuellen Falles zuordnen lassen, dass ihre Lösungen überhaupt für den aktuellen Fall nutzbar sind (siehe Abbildung 7.1). Die Teilmenge CB_{t_neu} der Case Base-Fälle, für die ein qualitativ zufriedenstellendes Mapping zum aktuellen Fall hergestellt werden kann, kann dann mit den bereits vorgestellten Ähnlichkeitsmaßen bearbeitet werden, um schließlich einen oder mehrere Fälle zu bestimmen, deren Attributkonstruktionen zur Lösung hinzugezogen werden.

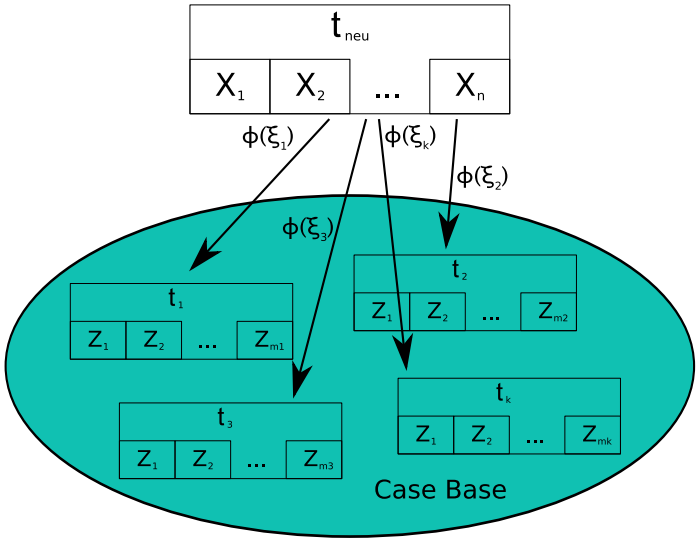


Abbildung 7.1.: Mapping der Attribute des neuen Falls auf die Attribute der Case Base-Fälle

7.2. Algorithmus

Um das oben beschriebene Problem der Attributzuordnung zu lösen, wird ein von Ingo Mierswa und dem Autor entworfener Algorithmus vorgestellt, der versucht, semantische Ähnlichkeiten zwischen den Basisattributen zu finden.

Der Begriff der (semantischen) Ähnlichkeit von Attributen soll anhand eines kurzen Beispieles umrissen werden: Er bezieht sich darauf, dass zwei Attribute die gleiche oder eine sehr ähnliche Größe wie z.B. Warenpreis, Entfernung, Luftdruck, etc. abbilden. Da in dieser Arbeit Datensätze behandelt werden, deren Attribute keinerlei sinntragende Benennung aufweisen müssen, ist die Möglichkeit, von Attributnamen auf solche Ähnlichkeiten zu schließen nicht gegeben. Daher wird hier versucht, aufgrund von noch näher zu erläuternden statistischen Merkmalen der Attributausprägungen auf Ähnlichkeiten zwischen den Attributen zu schließen.

Die diesem Ansatz zugrundeliegende Annahme ist die, dass zwei ähnliche Attribute X_i und Z_j ähnliche Wertemengen besitzen. Die Ähnlichkeit dieser Wertemengen wiederum wird anhand statistischer Merkmale wie z.B. der zugrundeliegenden Verteilung, dem Mittelwert oder der Standardabweichung geschätzt. Ein Beispiel für ein solches Attribut wäre z.B. die Körpergröße eines Erwachsenen in Metern, die einen Mittelwert von vielleicht 1,76 und eine Standardabweichung von etwa 0,08 haben könnte (Tabelle 7.1). Einem anderen Attribut, dessen Mittelwert z.B. bei 1,75 liegt und das eine Standardabweichung von 0,17 aufweist, kann mit einiger Wahrscheinlichkeit unterstellt werden, dass es sich bei seinen Ausprägungen ebenfalls um etwas handelt, dass Körpergrößen ähnelt. Diese Annahme muss natürlich nicht zutreffen. Ebenso problematisch wäre der Fall, dass ein Attribut Körpergrößen in Zentimeter und ein anderes in Metern mißt. Trotz dieser

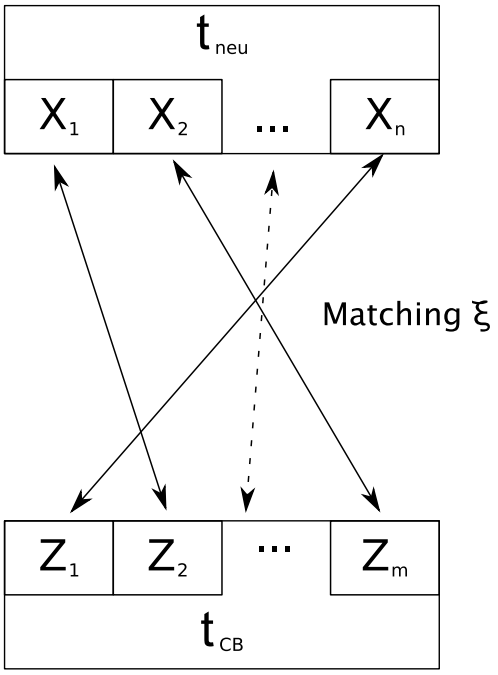


Abbildung 7.2.: Matching zwischen den Attributen zweier Fälle

Fehlerpotentiale soll die hier getroffene Annahme im Folgenden benutzt werden.

Im Weiteren wird zunächst dargestellt, wie die Ähnlichkeit zweier Attribute bestimmt wird, bevor schließlich der algorithmische Rahmen erläutert wird, der aus den Ähnlichkeiten der Attribute Matchings zwischen einem aktuellen Fall und allen Fällen in der Case Base herstellt. Der Pseudocode des Listings 7.1 gibt einen Überblick über den Gesamtablauf.

7.2.1. Goodness of Fit

Wie schon kurz umrissen, ist die erste Teilaufgabe des Algorithmus die Berechnung eines Ähnlichkeitswertes für zwei Attribute. Dieser Wert, der in der Fachliteratur häu-

Körpergröße	Gewicht	Schuhgröße
1,80	78	43
1,73	81	42
1,66	67	40
1,85	84	44
...	...	...

$\overset{?}{\Leftrightarrow}$

Att1	Att2	Att3
71	43	1,85
80	45	1,72
63	39	1,52
95	46	1,90
...	...	...

Tabelle 7.1.: Beispieldatensätze mit ähnlichen Attributen

Listing 7.1: Rahmenalgorithmus

```
1  matchCases(Case t_neu) {
2    foreach Case t ∈ CB{
3      foreach Attribute X_i ∈ t_neu {
4          foreach Attribute Z_j ∈ t {
5                  GoF_{i,j} = (X_i, Z_j);
6          }
7      }
8      sort(GoF_{i,j});
9      while not all X_i ∈ t_neu are matched {
10         Matching_{t,a,b} = select_best_Matching(GoF_{i,j});
11         delete_all_other_Matchings(a,b);
12     }
13   }
14 }
```

fig "Goodness of Fit" (GoF) genannt wird [Snedecor und Cochran 1989], beruht auf der Ähnlichkeit der Verteilungen zweier Wertmengen. Bei der Bestimmung dieser Ähnlichkeit können mehrere Ansätze verfolgt werden: Zum einen kann eine Grundverteilung (z.B. die Gaußverteilung) für beide Wertemengen angenommen werden und dann deren charakteristische Parameter (Mittelwert μ und Standardabweichung σ bei Gaußverteilung) für beide Mengen verglichen werden. Eine etwas genauere, aber dafür aufwendigere Möglichkeit besteht darin, die Überlappung der Dichtefunktionen beider Wertemengen zu bestimmen (siehe Beispiel in Abbildung 7.3). Bekannte Ansätze aus der Literatur sind etwa der χ^2 Goodness-of-Fit Test [Snedecor und Cochran 1989], der Kolmogorov-Smirnov Test [Chakravarti und Roy 1967] und der Anderson-Darling Test [Stephens 1974, Stephens 1976]. Leider sind für diese Tests entweder die vollständigen Wertemengen oder wenigstens eine größere Menge an Stichproben oder aggregierten Daten notwendig. Speicherplatz- und Rechenzeitbeschränkungen bei Aufbau und Verwaltung der Case Base lassen diese Tests daher als zu aufwendig erscheinen.

GoF1: Einfache Überdeckungsbestimmung

Um einen möglichst geringen Mehraufwand an Speicherplatz und Rechenzeit für die Bestimmung der Matchings zu gewährleisten, wird zunächst eine sehr simple Form des Goodness of Fit-Tests benutzt. Dabei wird für die Wertemengen der beiden zu vergleichenden Attribute X und Z jeweils angenommen, dass sie normalverteilt sind. Für beide Attribute werden die Intervalle I_X und I_Z bestimmt, in denen der Großteil der jeweiligen Wertausprägungen liegt. Bei einer Stichprobe über eine normalverteilte Zufallsvariable können ca. 95% aller Stichprobenwerte innerhalb des Intervalls von $[\mu-2\sigma, \mu+2\sigma]$ erwartet werden (vergleiche etwa [Sachs 1997]). Für die beiden so festgelegten Intervalle wird dann die Überschneidung und damit der GoF-Wert nach folgender Formel bestimmt:

$$GoF_1(X,Z) = \min_{i\in\{X,Z\}}(\mu_i + 2\sigma_i) - \max_{i\in\{X,Z\}}(\mu_i - 2\sigma_i),$$

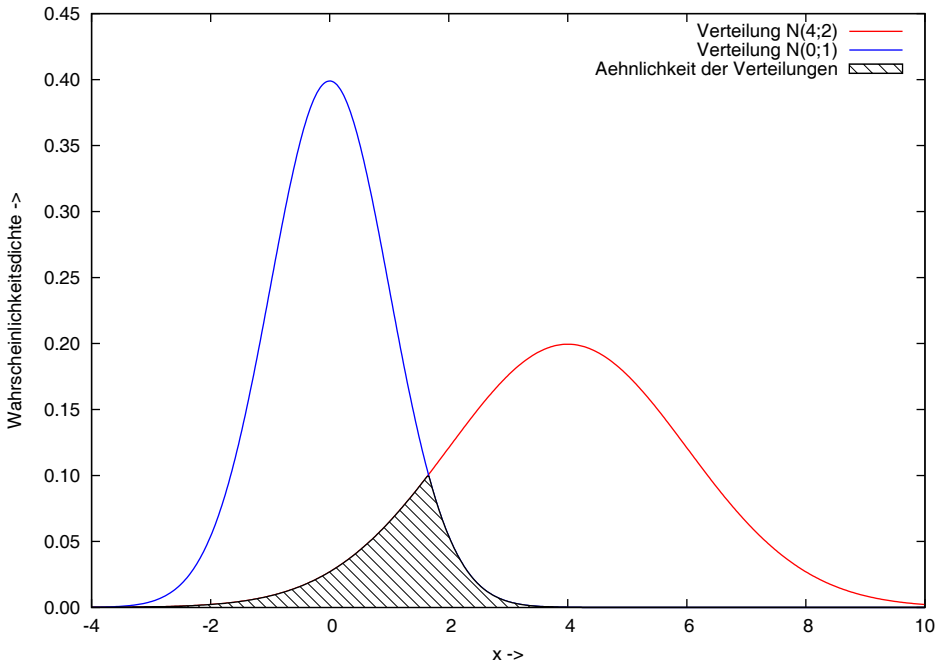


Abbildung 7.3.: Zwei Wahrscheinlichkeitsdichtefunktionen und ihre Überlappung

wobei μ den Mittelwert und σ die Standardabweichung der Stichprobenwerte bezeichnet. Zur Veranschaulichung der Intervallberechnung siehe Abbildung 7.4.

7.2.2. Attribut-Attribut-Mapping

Nachdem im vorherigen Abschnitt beschrieben wurde, wie der GoF-Wert zweier Attribute berechnet werden kann, wird jetzt erläutert, wie aus diesen Informationen ein komplettes Matching zwischen zwei Problemfällen t und t_{neu} hergestellt wird. Zunächst wird zwischen jedem Attribut $X_i \in t_{neu}$ und $Z_j \in t$ der GoF-Wert berechnet (Listing 7.1, Z. 3-7). Aus der dabei resultierenden Matrix $GoF_{i,j}$ wird dann im Stile eines Greedy-Algorithmus der beste Wert herausgenommen und die zugehörigen Attribute X_a und Z_b werden miteinander verbunden (Z. 10). Dann werden in $GoF_{i,j}$ die Zeile a und die Spalte b gelöscht, damit die Eindeutigkeit des Matchings gewährleistet bleibt (Z. 11). Dies wird solange wiederholt, bis alle Attribute $X_i \in t_{neu}$ auf ein Attribut aus t gematcht sind. Ein Beispiel soll diesen Ablauf verdeutlichen:

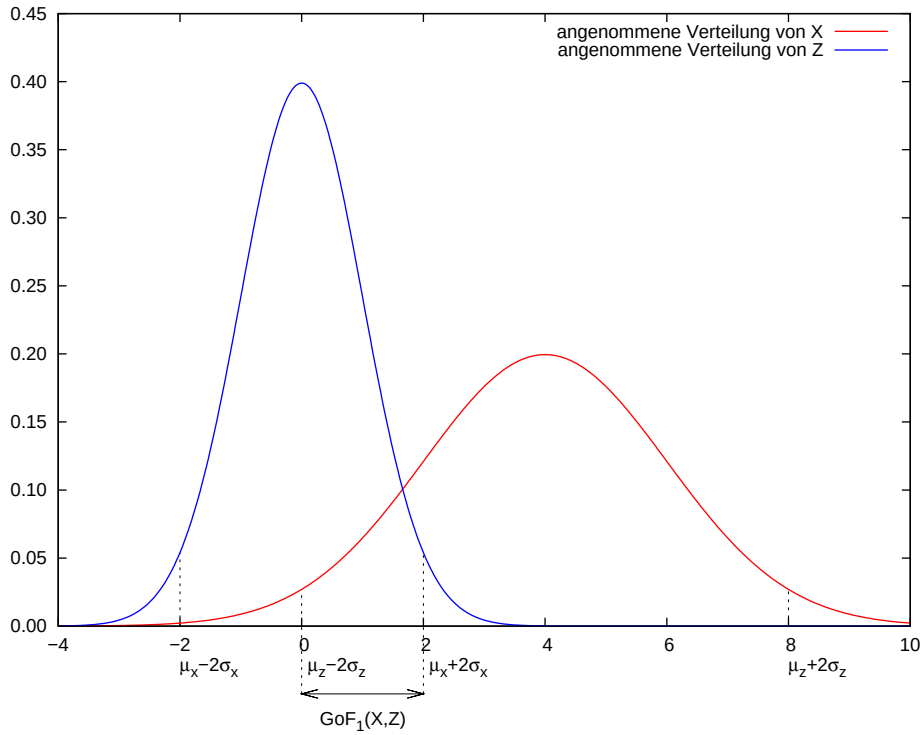


Abbildung 7.4.: Zwei Wahrscheinlichkeitsdichtefunktionen und ihre Intervallüberschneidung

$$(GoF_{i,j}) = \begin{pmatrix} - & Z_1 & Z_2 & Z_3 & Z_4 \\ X_1 & 0,7 & 1,2 & 9,3 & 0 \\ X_2 & 13,1 & 5,7 & 1,6 & 7,1 \\ X_3 & 0 & 0,2 & 3,8 & 12,9 \\ X_4 & 1,7 & \mathbf{18,3} & 0,1 & 10,8 \end{pmatrix}$$

$\Rightarrow$ 1. Verbindung: $X_4 \rightarrow Z_2$ Wert: 18,3

$$(GoF_{i,j}) = \begin{pmatrix} - & Z_1 & Z_3 & Z_4 \\ X_1 & 0,7 & 9,3 & 0 \\ X_2 & \mathbf{13,1} & 1,6 & 7,1 \\ X_3 & 0 & 3,8 & 12,9 \end{pmatrix}$$

$\Rightarrow$ 2. Verbindung: $X_2 \rightarrow Z_1$ Wert: 13,1

$$(GoF_{i,j}) = \begin{pmatrix} - & Z_3 & Z_4 \\ X_1 & 9,3 & 0 \\ X_3 & 3,8 & \mathbf{12,9} \end{pmatrix}$$

$\Rightarrow$ 3. Verbindung: $X_3 \rightarrow Z_4$ Wert: 12,9

$$(GoF_{i,j}) = \begin{pmatrix} - & Z_3 \\ X_1 & \mathbf{9,3} \end{pmatrix}$$

$\Rightarrow$ 4. Verbindung: $X_1 \rightarrow Z_3$ Wert: 9,3

Falls t weniger Attribute hat als t_{neu}, so bleiben die überzähligen Attribute aus t_{neu} unberücksichtigt beim Matching. Im umgekehrten Fall, wenn also t mehr Attribute hat als t_{neu}, muss dafür Sorge getragen werden, dass die nicht zugeordneten Attribute aus t keine Probleme verursachen, wenn die zu t gehörigen Merkmalskonstruktionen benutzt werden. Es ist dann darauf zu achten, dass die Merkmalskonstrukte gefiltert und nur diejenigen benutzt werden, in denen keine ungemappten Basisattribute vorkommen. Für diese Konstruktionen könnten für t_{neu} keine Instanzwerte berechnet werden, was zu Fehlern im Ablauf des benutzten Lernverfahrens führen würde.

7.2.3. Gesamtablauf

Die beiden vorangegangenen Abschnitte 7.2.1 und 7.2.2 beschreiben das Vorgehen des Algorithmus auf Attribut- und Fallebene. Um den Ablauf auf die gesamte Case Base auszudehnen, muss nur noch zwischen dem aktuellen Fall t_{neu} und jedem Fall $t \in CB$ ein Matching hergestellt werden. Um dann zu entscheiden, welche Case Base-Fälle das beste Matching zu t_{neu} aufweisen und damit t_{neu} am ähnlichsten erscheinen, werden wiederum die Gof-Werte herangezogen: Für jedes Matching ξ zwischen einem $t \in CB$ und t_{neu} wird die Summe der GoF-Werte der ausgewählten Attributzuordnungen gebildet und als Güte $\phi(\xi)$ benutzt. Für das Beispiel in Abschnitt 7.2.2 besteht das ausgewählte Matching ξ aus den Attributzuordnungen $X_4 \rightarrow Z_2$, $X_2 \rightarrow Z_1$, $X_3 \rightarrow Z_4$ und $X_1 \rightarrow Z_3$, die Güte ist $\phi(\xi) = 18,3 + 13,1 + 12,9 + 9,4 = 53,7$. Der Matchingalgorithmus kann dann entweder mit einem Grenzwert für ϕ parametrisiert werden, um eine Auswahl an Cases zu treffen, oder es wird ein fester Wert m vorgegeben, so dass die m am besten passenden Cases ausgewählt werden. Die so ausgewählte Teilmenge $CB_m \subseteq CB$ kann dann mit Hilfe der bereits beschriebenen Ähnlichkeitsmaße wie gewohnt nach passenden Cases durchsucht werden. Da die konstruierten Attribute dieser so ausgewählten Fälle wiederum Basisattribute beinhalten können (vergleiche z.B. Abbildung 6.1), müssen die Konstrukte umgebaut werden. Dabei werden die ursprünglichen Basisattribute von t durch die Basisattribute von t_{neu} gemäß ξ ersetzt. Ähnlich wird bei Distanzbestimmung über die SVM-Gewichtung verfahren, denn auch hierbei müssen natürlich die Attributnamen umgesetzt werden.

7.2.4. Laufzeit

Zum Abschluß soll noch kurz auf die Laufzeit eingegangen werden, die der Matchingalgorithmus benötigt. Zunächst ist festzustellen, dass die Berechnung aller Matchings für t_{neu}

pro Feature Generator-Lauf nur einmal durchgeführt werden muss. Die Laufzeit setzt sich aus folgenden Komponenten zusammen: Es müssen $c = |CB|$ viele Matchings bestimmt werden (von t_{neu} zu jedem $t \in CB$). Für jedes dieser Matchings zwischen t_{neu} und einem t wird für jede Kombination eines $X_i \in t_{neu}, i = 1..n$ mit einem $Z_j \in t, j = 1..m$ ein Goodness of Fit-Wert berechnet. Die GoF-Berechnung zweier Attribute wird mit Hilfe einer konstanten Anzahl arithmetischer Grundoperationen durchgeführt, so dass sich schließlich eine Gesamtlaufzeit von

$$O(c \cdot m \cdot n)$$

für den Matchingalgorithmus ergibt. Wie sich in den nachfolgend beschriebenen Experimenten hergestellt hat, werden in der Praxis ca. 1-3 Sekunden benötigt, um bei einer Case Base mit $c = 10000$ Fällen und im Durchschnitt 15 Basisattributen pro Fall alle Matchings zu berechnen. In Anbetracht der wesentlich längeren Gesamtlaufzeit eines Feature Generator-Laufes ist diese Laufzeit also zu vernachlässigen. Der Mehraufwand an Speicherplatz für die Case Base ist linear zu Anzahl der abgelegten Cases und ihrer Basisattribute: Für jedes dieser Attribute wird nun neben dem SVM-Gewicht noch der Mittelwert und die Standardabweichung seiner Merkmalsausprägungen gespeichert.

7.3. Experimente

Auch dieser Ansatz zur Case Base-unterstützten Feature Generation wird anhand einer Experimentreihe praktisch untersucht. Dabei werden im Wesentlichen die gleichen Experimentaufbauten wie in Kapitel 6 benutzt, die daher hier nicht mehr erläutert werden müssen. Da jedoch bei der Erzeugung der einzelnen Fälle sowie der Case Base einige Besonderheiten beachtet werden müssen, werden diese im folgenden Unterabschnitt genauer erläutert.

7.3.1. Vorbemerkungen

Da der gesamte Ansatz zum Matching von Basisattributen darauf basiert, dass Attribute mit gleicher Bedeutung eine ähnliche Werteverteilung aufweisen, müssen die synthetisch erzeugten Problemfälle in dieser Hinsicht bestimmten Ansprüchen genügen, um die Experimente sinnvoll durchzuführen. Dazu musste der in Kapitel 6 beschriebene Datensatzgenerator um zwei Fähigkeiten erweitert werden: Zum einen ist die Anzahl der Basisattribute randomisiert worden, zum anderen konnten jetzt die Werte der einzelnen Attribute normalverteilt statt wie vorher gleichverteilt gezogen werden, wobei Mittelwert und Standardabweichung wiederum randomisiert wurden. So hat der Matchingalgorithmus die Chance, überhaupt Unterschiede in der Ähnlichkeit verschiedener Attribute zu bestimmen. Sowohl die Fälle der Case Base mit 10000 Einträgen als auch die für die Experimente verwendeten Datensätze sind so entstanden. Die Attributanzahlen der zehn als neue Lernprobleme verwendeten Datensätze sind in Tabelle 7.2 aufgeführt.

7.3.2. Einmalige Erweiterung des Merkmalsraumes

Wie schon in Abschnitt 6.2 beschrieben wurden die 10 Datensätze in der Testreihe 4a ohne Benutzung eines evolutionären Feature Generators an einen linearen Lernoperator

Datensatz Nr.	Anzahl Basisattribute
1	5
2	2
3	12
4	8
5	11
6	10
7	2
8	11
9	5
10	7

Tabelle 7.2.: Anzahl der Basisattribute der Testfälle

übergeben. Zuvor wurden die Datensätze um die konstruierten Attribute von 1, 10 bzw. 40 Fällen aus der Case Base erweitert. Zum Vergleich wurde auch wieder eine Experimentreihe ohne Benutzung von abgelegten Cases durchgeführt, deren Ergebnisse in Tabelle 7.3 in der Spalte "Lin. Regression" aufgeführt sind. Abbildung 7.5 zeigt die Ergebnisse als Balkendiagramme für die einzelnen Datensätze. Der Performanzgewinn steigt merklich an mit der Anzahl der benutzten Cases. Während die Ergebnisse für die Experimentreihe mit nur einem Case im Mittel nicht signifikant von denen der einfachen linearen Regression abweichen, zeigt eine Varianzanalyse der Ergebnisse des linearen Lerners und der Ergebnisse der Experimentreihe mit 40 Cases einen Unterschied auf fünfprozentigem Signifikanzniveau.

7.3.3. Case Base-Unterstützung eines evolutionären Feature Generators

Auch die zweite Anwendungmöglichkeit des Case Base-Ansatzes, nämlich die im Rahmen eines Feature Generators, wurde unter Verwendung des Matchingalgorithmus praktisch erprobt. Bei den im Folgenden beschriebenen Experimenten kam es der Laufzeit zugute, dass die Matchings zwischen einem neuen Fall und einer Case Base nur einmalig berechnet werden mussten und danach für jede weitere Case Base-Anfrage wiederverwendet werden konnten. Für diese Experimente wurden dieselben Testdatensätze und dieselbe Case Base verwendet wie für die Testreihen in Abschnitt 7.3.2, so dass die Vergleichbarkeit der Ergebnisse gewährleistet ist.

Die 10 Testdatensätze wurden jeweils einmal mit dem Case Base-unterstützten Yagga3-Operator unter Verwendung des Matchingalgorithmus und einmal mit dem klassischen Featuregenerator Yagga2 durchgeführt. Ein fairer Vergleich mit dem Yagga3-Operator ohne die Fähigkeit zum Attributmatching (siehe Kapitel 6) konnte nicht durchgeführt werden. Dies ist darin begründet, dass für diese Version des Yagga3-Operators sowohl die Fälle in der Case Base als auch die Testfälle alle die gleiche Attributanzahl haben müssen.

Nr.	Lin. Regression	1 Case	10 Cases	40 Cases
1	1,0179	1,009	1,006	0,1169
2	0,9847	0,5798	0,2034	0,0965
3	0,9855	0,9916	0,6108	0,2304
4	0,2492	0,25	0,0518	0,0438
5	0,2958	0,2899	0,2287	0,0296
6	0,57	0,3704	0,2758	0,2489
7	0,9917	0,8618	0,4071	0,0580
8	0,9992	1,0106	0,5828	0,0554
9	1,026	1,0084	0,7405	0,7432
10	1,0083	0,8766	0,5347	0,5236
∅/Std.Abw.	0,8128/0,32	0,7248/0,32	0,4642/0,29	0,2146/0,24

Tabelle 7.3.: Testreihe 4a: Einmalige Erweiterung des Merkmalsraumes um die Attributkonstruktionen von 1, 10 und 40 vorgeschlagenen Case Base-Fällen. Zum Vergleich Ergebnisse des linearen Lerners ohne Case Base-Unterstützung. Performanzmaß: Root Mean Squared Error

Beschränkung der Generationenzahl

Zunächst wurden zwei Reihen von Feature Generator-Läufen mit einer festen Obergrenze von 20 Generationen durchgeführt. Bei den Experimenten mit dem Yagga3-Operator wurde beginnend mit der zweiten Generation alle 5 Generationen Case Base-Aufrufe durchgeführt mit einer Wahrscheinlichkeit von 50% pro Individuum. Die übrigen Mutationsoperatoren hatten ebenso wie beim Yagga2 die voreingestellten Wahrscheinlichkeiten von ebenfalls 50%.

Die Einzelergebnisse sind in Tabelle 7.4 und Abbildung 7.6 dargestellt. Der Performanzgewinn fällt sehr unterschiedlich aus, ist jedoch in allen Fällen deutlich zu erkennen. Es kann hier per Varianzanalyse festgestellt werden, dass sich die Ergebnisse im Mittel auf dem zehnprozentigen Signifikanzniveau unterscheiden.

Beschränkung der Laufzeit

Zum Abschluß der Experimente zum Matching-Ansatz wurde ein weiterer Feature Generator-Lauf durchgeführt, diesmal mit einer auf 100 Sekunden pro Testfall begrenzten Laufzeit. Diese Experimentreihe soll unter praktischen und fairen Bedingungen zeigen, ob der Case Base-Ansatz selbst unter der erschwerten Voraussetzung fehlender Attributzuordnungen bei gleicher Laufzeit einen Vorteil gegenüber der herkömmlichen Feature Generation einbringt.

Auch hier fiel der Performanzunterschied bei den einzelnen Problemfällen verschieden aus. Während bei Datensatz Nr. 3 vom Yagga3 zwar ein leicht schlechteres Ergebnis erzielt wurde als vom Yagga2, so zeigt Abbildung 7.7 jedoch, dass in den anderen Fällen ein deutlicher Performanzgewinn festgestellt werden konnte.

Nr.	Yagga2	Yagga3
1	0,2231	0,1482
2	0,5621	0,029
3	0,8409	0,002
4	0,2438	0,039
5	0,2846	0,0364
6	0,4669	0,1748
7	0,8128	0,0334
8	0,7412	0,055
9	0,7285	0,694
10	0,5920	0,4274
∅/Std.Abw.	0,5496/0,24	0,1639/0,23

Tabelle 7.4.: Testreihe 4b: Generationenbeschränkte Läufe der Feature Generatoren Yagga2 und Yagga3. 20 Generationen pro Testdatensatz. Performanzmaß: Root Mean Squared Error

Nr.	Yagga2 100 Sek.	Yagga3 100 Sek.
1	0,9932	0,1822
2	0,5647	0,0778
3	0,5414	0,6332
4	0,1418	0,0438
5	0,0458	0,0339
6	0,3843	0,2467
7	0,8052	0,0422
8	0,689	0,0844
9	0,7772	0,7158
10	1,0002	0,4515
∅/Std.Abw.	0,5943/0,33	0,2512/0,26

Tabelle 7.5.: Testreihe 4c: Zeitbeschränkte Läufe der Feature Generatoren Yagga2 und Yagga3. 100 Sekunden pro Testdatensatz. Performanzmaß: Root Mean Squared Error

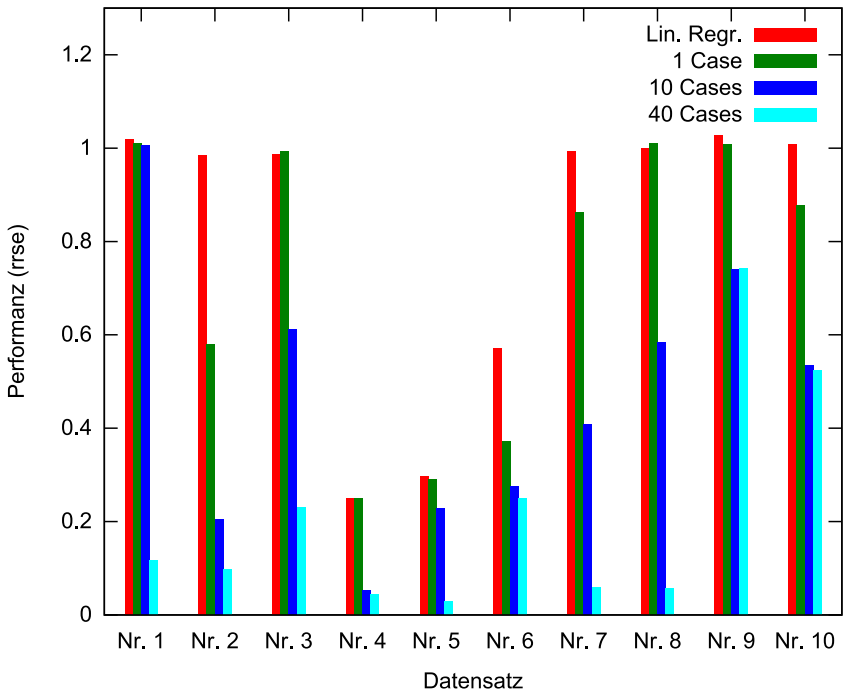


Abbildung 7.5.: Testreihe 4a: Einmalige Erweiterung des Merkmalsraumes um die Attributkonstruktionen von 1, 10 und 40 vorgeschlagenen Case Base-Fällen. Zum Vergleich Ergebnisse des linearen Lerners ohne Case Base-Unterstützung.

7.4. Fazit

Dieser erste, relativ einfache Algorithmus zum Attributmatching zeigt bereits, dass in dieser Methode das Potential steckt, den Case Base-Ansatz in der Praxis anwendbar zu machen. Er bietet die Möglichkeit, abgelegtes Lösungswissen einer ganzen Klasse von Lernproblemen zu verwenden, unabhängig vom Anwendungsbereich eines Datensatzes oder der konkreten Anzahl und Reihenfolge seiner Basisattribute. Die durchgeführten Experimente haben vielversprechende Ergebnisse erbracht, die eine weitere Beschäftigung mit dem gesamten Case Base-Ansatz sinnvoll erscheinen lassen.

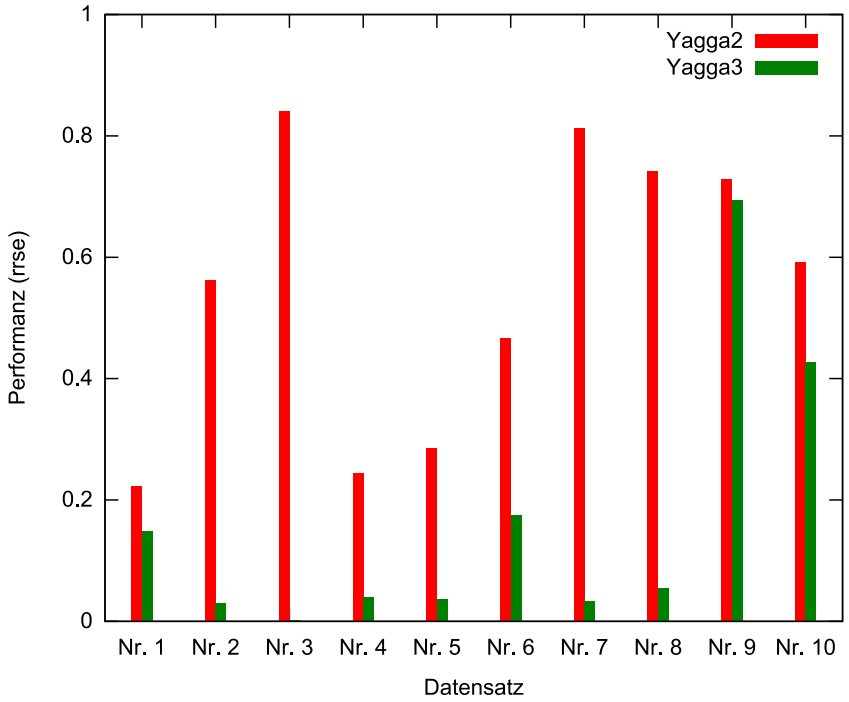


Abbildung 7.6.: Testreihe 4b: Beschränkung auf 20 Generationen (10 Individuen), Performanzen von Yagga2, Yagga3

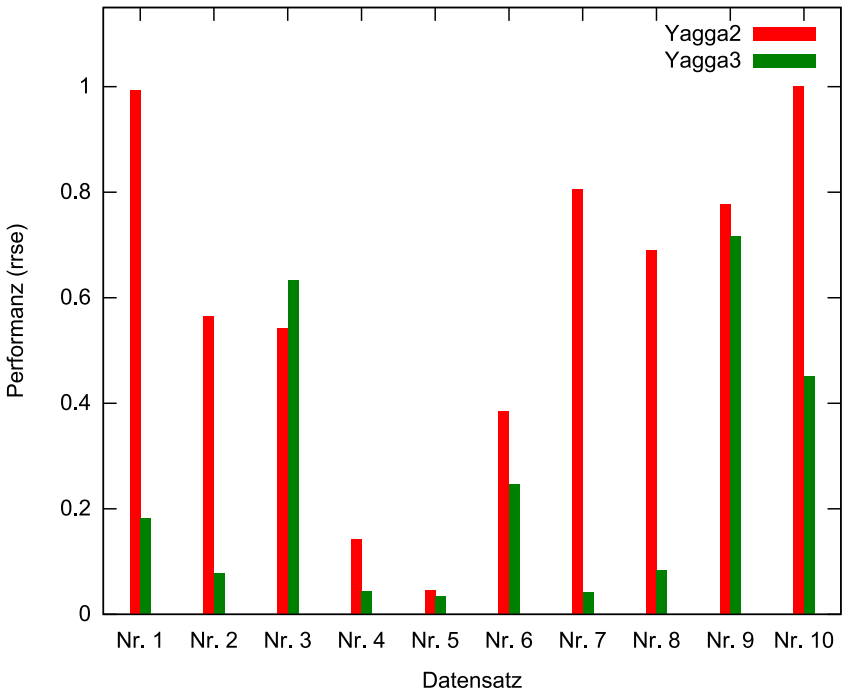


Abbildung 7.7.: Testreihe 4c: Beschränkung auf 100 Sekunden Laufzeit pro Fall, Performanzen von Yagga2, Yagga3

8. Zusammenfassung

Die Entdeckung komplexer funktionaler Zusammenhänge zwischen Basisattributen und Zielmerkmal eines Datensatzes überfordert im Allgemeinen einfache Methoden wie die lineare Regression. Zwar gibt es nicht-lineare Lernverfahren wie künstliche neuronale Netze oder die kernelbasierte SVM, diese jedoch erzeugen Modelle, die für einen menschlichen Betrachter nicht mehr nachvollziehbar sind. Einen Ausweg, der sowohl eine gute Regressionsperformanz als auch ein verständliches Modell des Funktionszusammenhangs liefert, besteht in der Benutzung der Merkmalskonstruktion als Meta-Lernverfahren. Zur Unterstützung und Beschleunigung dieses Verfahrens wurde ein dem Prinzip des fallbasierten Schließens nachempfundener Ansatz entwickelt und erprobt.

8.1. Rückblick

Basierend auf einer Idee von [Mierswa und Wurst 2005b] wurden zuerst die theoretischen Grundlagen des Case Base-Ansatzes untersucht. Dabei ging es zunächst um die Repräsentation von Datensätzen durch einen Gewichtsvektor. Die dabei durchzuführende Komprimierung der zu speichernden Datensätze ermöglicht einerseits den Speicherplatz sparenden Aufbau großer Case Bases, andererseits bildet sie die Basis der effizienten Suche nach möglichst ähnlichen Lernproblemen. Im Rahmen dieser Arbeit konnte dabei ein etwas ausführlicherer Teilbeweis für die Eignung der SVM-Gewichtung geführt werden.

Der nächste Schritt bestand dann in der Betrachtung verschiedener Metriken und Ähnlichkeitsmaße, mit denen auf Basis der Gewichtsvektoren gleichartige Lernprobleme identifiziert werden können. Dabei wurden eine Vielzahl von Maßen auf ihre Eignung bzgl. vorher definierter Kriterien untersucht. Dies geschah teilweise mit Hilfe von Beweisen, teilweise empirisch. Es konnte ein weiteres Maß entdeckt werden, das zumindest die wichtigsten Kriterien erfüllt. Zusätzlich zur gewichtungsbasierten Abstandsmessung wurde dann das samplingbasierte Ähnlichkeitsmaß eingeführt, das eine feinere Auswahl der Case Base-Antworten im Rahmen eines Feature Generator-Laufes ermöglicht.

Für die praktische Evaluierung wurde ein Plugin für das Data Mining-Werkzeug RapidMiner entwickelt, welches den Case Base-Ansatz implementiert. Die damit durchgeführten Experimente zeigten vielversprechende Ergebnisse, die darauf schließen lassen, dass sich der Case Base-Ansatz sowohl zur Beschleunigung der Feature Construction als auch zur Verbesserung der einfachen linearen Regression einsetzen läßt. Schließlich konnte der Grundansatz in Kapitel 7 so erweitert werden, dass Fälle mit verschiedenen Basisattributen in einer Case Base zusammengefaßt und benutzt werden können. Diese Fähigkeit läßt einen praktischen Einsatz des Verfahrens in greifbare Nähe rücken. Dies gilt umso mehr, als die Möglichkeit besteht, mit der in Abschnitt 6.1.2 erläuterten Methode zur Erzegung von "synthetischen" Case Base-Einträgen eine große und universell einsetzbare Fallbasis mit relativ geringem Rechenaufwand zu erzeugen.

8.2. Kritische Bewertung

Der vorgestellte Ansatz hat, wie in den Experimenten gezeigt wurde, durchaus das Potential, Lernverfahren wie die lineare Regression oder Feature Construction wirkungsvoll zu unterstützen. Es sind jedoch einige Voraussetzungen zu beachten. Zum einen ist der Case Base-Ansatz in seiner hier erarbeiteten Form nur für die Regression bzw. Klassifikation auf numerischen, gelabeltet Datensätzen anzuwenden. Die Einbeziehung nominaler Attribute etwa ist nicht ohne weiteres möglich, hier müßte der Benutzer entweder diese Attribute aus dem Datensatz herausnehmen oder sie geeignet umkodieren. Zum anderen ist zu bedenken, dass sich der Ansatz auf mehreren Ebenen auf Heuristiken verläßt, die als solche natürlich nicht unfehlbar sind. Da wäre zum einen die Grundannahme, dass die Relevanz der Basisattribute mit den für einen Datensatz konstruierten Features korreliert (siehe Aschnitt 2.6). Eine weitere Ebene, auf welcher heuristisch gearbeitet wird, ist die Zuordnung der Basisattribute (Feature Matching, siehe Kapitel 7).

Die Ergebnisse der in dieser Arbeit durchgeführten Experimente deuten jedoch darauf hin, dass der Case Base-Ansatz trotz der eben erwähnten Einschränkungen eine wirkungsvolle Verbesserung vorhandener Lernverfahren darstellt. Diese Verbesserung zeigt sich auf zwei Arten:

- Die Abkürzung langdauernder Lernverfahren wie der evolutionären Feature Construction durch Konvergenzbeschleunigung und
- die Verbesserung der Regressionsperformanz schneller und einfacher Verfahren wie der linearen Regression, wenn auch unter erhöhtem Laufzeitaufwand.

Insgesamt bietet der Case Base-Ansatz in den vorgestellten Anwendungsszenarios also einen echten Vorteil. Die Abwägung zwischen Laufzeit und Regressionsperformanz kann dabei sehr flexibel gehandhabt werden, in dem der Benutzer einfach die Menge der von der Case Base zurückgelieferten Merkmalskonstruktionen anpasst.

8.3. Ausblick

Trotz aller positiven Ergebnisse bleiben noch verschiedene Punkte offen, die Anlass zu weiteren Untersuchungen geben können.

- **Optimierung der Implementierung:** Die für diese Arbeit benutzte Implementierung lädt die Case Base komplett in den Hauptspeicher, was für große Case Bases ($>$ 1.000.000 Einträge) nicht mehr tragbar ist. Hier wäre die Entwicklung einer datenbankgestützten Lösung eine Aufgabe für die Zukunft.
- **Verteilte Architektur:** Im Rahmen der immer weiter fortschreitenden Vernetzung und Dezentralisierung von Rechenleistung ist die gemeinsame Nutzung einer Case Base durch mehrere Benutzer ebenfalls eine interessante Erweiterungsmöglichkeit, die den Nutzen des CBR-Ansatzes deutlich erhöht. Aufgrund der äußerst kompakten Repräsentation der Lernprobleme kann eine Anfrage an die Case Base auch in bandbreitenbeschränkten Einsatzszenarien sehr effizient durchgeführt werden.

- **Goodness of Fit:** Die bisher benutzte Funktion zur Bestimmung des GoF-Wertes ist recht simpel gewählt. Dies hat zwar den nicht unwichtigen Vorteil der schnellen Berechenbarkeit, dennoch sind an dieser Stelle noch Verbesserungsmöglichkeiten denkbar, von denen einige schon in Abschnitt 7.2.1 kurz angesprochen wurden.
- **Erzeugung der Case Base:** In dieser Arbeit wurden zur Erzeugung der Fälle für die Case Bases stets zufällig generierte Funktionen benutzt, da diese schnell und in beinahe beliebiger Menge zur Verfügung standen (siehe Abschnitt 6.1.2). Es bleibt zu erproben, ob die Benutzung von realen (also durch Messungen oder Umfragen etc. gewonnenen) Datensätzen und der für sie konstruierten Features überhaupt lohnt, oder ob der drastisch erhöhte Aufwand für Beschaffung und Bearbeitung solcher Datensätze vielleicht zu gar keiner nennenswerten Verbesserung der Case Base-Qualität führen würde.

Literaturverzeichnis

[Abraham 2003] Abraham, Ajith (2003). *Meta-Learning Evolutionary Artificial Neural Networks*.

[Banzhaf et al. 1998] Banzhaf, W., P. Nordin, R. Keller und F. Francone (1998). *Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and its Applications*. Morgan Kaufmann.

[Brazdil et al. 2003] Brazdil, Pavel B., C. Soares und J. P. D. Costa (2003). *Ranking Learning Algorithms: Using IBL and Meta-Learning on Accuracy and Time Results*. Mach. Learn., 50(3):251–277.

[Breiman et al. 1984] Breiman, L., J. H. Friedman, R. A. Olshen und C. J. Stone (1984). *Classification and Regression Trees*. Wadsworth, Belmont, Calif.

[Chakravarti und Roy 1967] Chakravarti, Laha und Roy (1967). *Handbook of Methods of Applied Statistics*, Bd. Volume I. John Wiley and Sons.

[Ciaccia et al. 1997] Ciaccia, Paolo, M. Patella und P. Zezula (1997). *M-tree: An Efficient Access Method for Similarity Search in Metric Spaces.*. In: *VLDB*, S. 426–435.

[Cicirello 2000] Cicirello, Stephen F. Smith Vincent A. (2000). *Modeling GA Performance for Control Parameter Optimization*. In: Whitley, Darrell, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee und H.-G. Beyer, Hrsg.: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, S. 235–242, Las Vegas, Nevada, USA. Morgan Kaufmann.

[Cohen et al. 2002] Cohen, I., Q. Tian, X. Zhou und T. Huang (2002). *Feature Selection Using Principal Feature Analysis*.

[Dice 1945] Dice, L.R. (1945). *Measures of the Amount of Ecologic Association between Species*. Ecology, 26:297–302.

[Draper und Smith 1966] Draper, N. R. und H. Smith (1966). *Applied Regression Analysis*. Wiley, New York.

[Dörfler und Peschek 1988] Dörfler, Willibald und W. Peschek (1988). *Einführung in die Mathematik für Informatiker*. Carl Hanser Verlag München Wien.

[Drucker et al. 1997] Drucker, Harris, C. J. C. Burges, L. Kaufman, A. Smola und V. Vapnik (1997). *Support Vector Regression Machines*. In: Mozer, Michael C., M. I. Jordan und T. Petsche, Hrsg.: *Advances in Neural Information Processing Systems*, Bd. 9, S. 155. The MIT Press.

[Dunteman 1989] Dunteman, G.H. (1989). *Principal Components Analysis*. Sage Publications, Inc., Newbury Park, CA.

[Emran und Ye 2001] Emran, Syed Masum und N. Ye (2001). *Robustness of Canberra Metric in Computer Intrusion Detection*. In: *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security United States Military Academy, West Point, NY*.

[Guyon und Elisseeff 2003] Guyon, Isabelle und A. Elisseeff (2003). *An Introduction to Variable and Feature Selection*. J. Mach. Learn. Res., 3:1157–1182.

[Hecht-Nielsen 1990] Hecht-Nielsen, Robert (1990). *Neurocomputing*. Addison-Wesley.

[Jain et al. 1999] Jain, A. K., M. N. Murty und P. J. Flynn (1999). *Data Clustering: a Review*. ACM Computing Surveys, 31(3):264–323.

[Kira und Rendell 1992] Kira, K. und L. Rendell (1992). *The Feature Selection Problem: Traditional Methods an a New Algorithm*. In: *The Proceedings of the Tenth National Conference on Artificial Intelligence*.

[kopf et al. 1999] kopf, S., B. Mika, S. Burges, C. Knirsch, P. Miiller, K. itsch und G. Smola (1999). *Input Space vs. Feature Space in Kernel-based Methods*.

[Koza 1996] Koza, John R. (1996). *Genetic Programming*. MIT Press, Cambridge.

[Kreyszig 1975] Kreyszig, Erwin (1975). *Statistische Methoden und ihre Anwendungen*. Vandenhoek & Ruprecht, 5. Aufl.

[Kürsten 2006] Kürsten, Jens (2006). *Systematisierung und Evaluierung von Clustering-Verfahren im Information Retrieval*. Diplomarbeit, Technische Universität Chemnitz.

[Liu et al. 2002] Liu, H., H. Motoda und L. Yu (2002). *Feature Selection with Selective Sampling*. In: *Proceedings of the Nineteenth International Conference on Machine Learning*, S. 395 – 402.

[Mahalanobis 1936] Mahalanobis, P.C. (1936). *On the generalized Distance in Statistics*. In: *Proceedings of the National Institute of Science of India*, Bd. 2, S. 49–55. National Institute of Siences of India.

[McEnery et al. 1994] McEnery, Anthony M., M. P. Oakes und R. G. Garside (1994). *The Use of Approximate String Matching Techniques in the Alignment of Sentences in Parallel Corpora*. Technischer Bericht, The University of Lancaster, UK.

[Mierswa 2006] Mierswa, Ingo (2006). *Evolutionary Learning with Kernels: A Generic Solution for Large Margin Problems*. In: *Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2006)*. accepted for publication.

[Mierswa und Wurst 2005a] Mierswa, Ingo und M. Wurst (2005a). *Efficient Case Based Feature Construction for Heterogeneous Learning Tasks*. In: al., J. Gama et, Hrsg.: *Proc. of the European Conference on Machine Learning (ECML 2005)*, LNAI 3720, S. 641–648. Springer.

[Mierswa und Wurst 2005b] Mierswa, Ingo und M. Wurst (2005b). *Efficient Feature Construction by Meta Learning – Guiding the Search in Meta Hypothesis Space*. In: *Proc. of the International Conference on Machine Learning, Workshop on Meta Learning*.

[Mierswa et al. 2006] Mierswa, Ingo, M. Wurst, R. Klinkenberg, M. Scholz und T. Euler (2006). *YALE: Rapid Prototyping for Complex Data Mining Tasks*. In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006)*. ACM Press.

[Morariu et al. 2006] Morariu, Daniel, L. Vintan und V. Tresp (2006). *Evolutionary Feature Selection for Text Documents using the SVM*, Bd. 15 d. Reihe *TRANSACTIONS ON ENGINEERING, COMPUTING AND TECHNOLOGY*, S. 215–221. WORLD ENFORMATIKA SOCIETY. ISSN: 1305-5313.

[Myers und Rabiner 1981] Myers, C. und L. Rabiner (1981). *A Comparative Study of Several Dynamic Time-warping Algorithms for Connected Word Recognition*. The Bell System Technical Journal, 60(7):1389–1409.

[Quinlan 1993] Quinlan, J. Ross (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Quinlan 1986] Quinlan, R.J. (1986). *Induction of Decision Trees*. Machine Learning, 1(1):81–106.

[Rencher 1998] Rencher, Alvin C. (1998). *Multivariate Statistical Inference and Applications*. John Wiley, New York.

[Richardson 1968] Richardson, D. (1968). *Some Unsolvable Problems Involving Elementary Functions of a Real Variable*. Journal of Symbolic Logic, 33:514–520.

[Ritthoff et al. 2002] Ritthoff, Oliver, R. Klinkenberg, S. Fischer und I. Mierswa (2002). *A Hybrid Approach to Feature Selection and Generation Using an Evolutionary Algorithm*. Technischer Bericht CI-127/02, Collaborative Research Center 531, University of Dortmund, Dortmund, Germany. ISSN 1433-3325.

[Rodríguez 2004] Rodríguez, Carlos C. (2004). *The Kernel Trick*. http://omega.albany.edu:8008/machine-learning-dir/notes-dir/ker1/ker1-l.html downloaded Nov. 14th, 2006.

[Sachs 1997] Sachs, Lothar (1997). *Angewandte Statistik*. Springer, 8. Auflage Aufl.

[Smola und Schölkopf 2003] Smola, Alex J. und B. Schölkopf (2003). *A Tutorial on Support Vector Regression*. Technischer Bericht, NeuroCOLT2 Technical Report Series.

[Snedecor und Cochran 1989] Snedecor, George W. und W. G. Cochran (1989). *Statistical Methods*. Iowa State University Press, Eighth Edition Aufl.

[Stephens 1974] Stephens, M. A. (1974). *EDF Statistics for Goodness of Fit and Some Comparisons*. Journal of the American Statistical Association, Vol. 69:pp. 730–737.

[Stephens 1976] Stephens, M. A. (1976). *Asymptotic Results for Goodness-of-Fit Statistics with Unknown Parameters*. Annals of Statistics, Vol. 4:pp. 357–369.

[Thrun und O'Sullivan 1996] Thrun, S. und J. O'Sullivan (1996). *Discovering Structure in Multiple Learning Tasks: The TC Algorithm*. In: Saitta, L., Hrsg.: *Proceedings of the 13th International Conference on Machine Learning ICML-96*, San Mateo, CA. Morgen Kaufmann.

[Vapnik 1995] Vapnik, Vladimir N. (1995). *The Nature of Statistical Learning Theory*. Springer, New York.

[Wang und Witten 1999] Wang, Yong und I. H. Witten (1999). *Pace Regression*.

[Witten und Frank 2005] Witten, Ian H. und E. Frank (2005). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, 2nd. Aufl.

[Yang und Honavar 1997] Yang, Jihoon und V. Honavar (1997). *Feature Subset Selection Using A Genetic Algorithm*. In: Koza, John R., K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba und R. L. Riolo, Hrsg.: *Genetic Programming 1997: Proceedings of the Second Annual Conference*, S. 380, Stanford University, CA, USA. Morgan Kaufmann.

[Zhang et al. 1995] Zhang, K., J. T. L. Wang und D. Shasha (1995). *On the Editing Distance Between Undirected Acyclic Graphs and Related Problems*. In: *Proc. of the 6th Annual Symposium on Combinatorial Pattern Matching*, S. 395 – 407. Springer, Berlin.