Projektgruppe MEDMINER — Implementierung eines Data Mining-Baukastens für das maschinelle Lernen in der Intensivmedizin

Jens Fisseler*

*Universität Dortmund, Fachbereich Informatik, Email fissel00@marvin.cs.uni-dortmund.de

Zusammenfassung. Wissensentdeckung ist ein Prozess, der intensive Vorarbeiten und ein gut geplantes Vorgehen erfordert. Die Daten, in denen Wissen entdeckt werden soll, müssen sorgfältig ausgewählt und vorverarbeitet werden, wobei die Vorverarbeitung bei Verwendung unterschiedlicher Lernverfahren für alle Lernverfahren identische, aber auch unterschiedliche Vorverarbeitungsschritte enthält.
Dieser Artikel beschreibt die Arbeit der im Wintersemester 2000/2001 und Sommersemester 2001 an der Universität Dortmund stattgefundenen Projektgruppe MEDMINER. Aufgabe der Projektgruppe war das Lernen auf intensivmedizinischen Daten und das Entwickeln eines Data Mining-Baukastens. Da die Daten als Zeitreihen vorlagen, ergaben sich daraus schon eine Reihe von Problemen für die Vorverarbeitung. Wie diese Probleme gelöst wurden und wie ein Data Mining-Baukasten, kombiniert mit dem Einsatz von Metadaten, die Durchführung von Data Mining-Aufgaben erleichtern kann, wird in diesem Artikel geschildert.

Schlüsselwörter. Data Mining, Baukasten, Metadaten, Intensivmedizin

1 Einleitung

An der Universität Dortmund muss jeder Informatikstudent im Hauptstudium an einer sogenannten Projektgruppe teilnehmen, eine Lehrveranstaltung, die zwei Semester mit einem Gesamtaufwand von 16 SWS umfasst. In einer Projektgruppe sollen 8-12 Studenten gemeinsam an einem größeren Softwareprojekt arbeiten und dabei sowohl wissenschaftlich arbeiten als auch die Zusammenarbeit in einer Gruppe üben.

Die Projektgruppe MEDMINER fand im Wintersemester 2000/2001 und Sommersemster 2001 am Lehrstuhl für Künstliche Intelligenz des Fachbereichs Informatik der Universität Dortmund statt. Die Aufgabenstellung der Projektgruppe war zweigeteilt: Zum einen sollte Data Mining auf intensivmedizinischen Patientendaten betrieben werden, und zum anderen sollte ein Data Mining-Baukasten entworfen und implementiert werden, mit dessen Hilfe die gestellte und mögliche weitere Data Mining-Aufgaben effizient durchgeführt werden können.

Dieser Artikel soll die Arbeit der Projektgruppe MEDMINER vorstellen, wobei der Schwerpunkt nicht auf den Ergebnissen des Data Minings liegt, sondern auf der Implementierung des dafür implementierten Data Mining-Baukastens. In Abschnitt 2 werden dennoch kurz die Datensituation in der Intensivmedizin und mögliche Aufgaben für Data Mining-Experimente vorgestellt, da diese Einfluss auf die implementierten Komponenten des Baukastens hatten. In Abschnitt 3 wird dann der Data Mining-Baukasten vorgestellt. Zuerst werden seine einzelnen Komponenten (3.1) und deren Zusammenspiel beschrieben (3.2). In Abschnitt 3.3 wird dann die Durchführung eines Data Mining-Versuchs mit dem Baukasten skizziert, bevor in 3.4 die Vorteile eines Data Mining-Baukastens erläutert werden. Der Artikel schließt mit einer Diskussion über mögliche Erweiterungen des Baukastens.

2 Patientendaten und Lernaufgaben

Die intensivmedizinischen Patientendaten [4] wurden der Projektgruppe von Dr. Imhoff, einem Arzt der Städtischen Kliniken Dortmund, zur Verfügung gestellt. Die Rohdaten liegen als CSV[1]-Textdatei vor und umfassen 175 minütlich aufgezeichnete Variablen zu über 400 Intensivpatienten. Aufgezeichnet wurden Vitalwerte wie z.B. verschiedene Blutdrücke, die Herzfrequenz und daraus berechnete medizinische Indizes, Beatmungsparameter, demographische Daten, Dosierungen und Dosierungsänderungen von

[1] *c*omma separated *v*alues

Dauerinfusionen, Dosierungen von Bolusgaben[2], Ein-/Ausfuhrsummen und deren einzelne Komponenten sowie Laborwerte. Obwohl minütlich ein Datentupel aufgezeichnet wurde, wurden dennoch nicht alle Variablen minütlich gemessen. Laborwerte wurden z.B. nur in unregelmäßigen Abständen bestimmt, während Variablen die über Sensoren erfasst werden, z.B. die unterschiedlichen Blutdrücke und die Herzfrequenz, minütlich aufgezeichnet wurden. Trotz der unterschiedlichen zeitlichen Granularität, mit der die einzelnen Variablen aufgezeichnet wurden, liegen alle Variablen als Zeitreihen vor. Diesem Umstand muss die gesamte Vorverarbeitung Rechnung tragen, doch dazu wird in Abschnitt 3 mehr gesagt.

Folgende Lernaufgaben sind bei den vorliegenden intensivmedizinischen Verlaufsdaten von Interesse:

Alarmfunktion: Ist der Zustand eines Patienten stabil oder benötigt er eine medizinische Intervention, z.B. eine Medikamentengabe? Von Nutzen wäre eine „intelligente“ Alarmfunktion, die schon bei verdächtigen Tendenzen im Verlauf einiger Variablen den Arzt alarmiert.

Interventionsvorhersage: Welche Intervention sollte bei einem Patienten vorgenommen werden, das heißt z.B. welches Medikament sollte ihm verabreicht werden?

Gruppierung: Lassen sich Patienten anhand ihrer Reaktion auf Interventionen gruppieren? Gibt es auffällige Gruppen von Patienten, die besonders, d.h. nach einem anderen medizinischen Protokoll als die restlichen Patienten, behandelt werden?

Validierung: Reagiert der Patient auf die vorgenommene Intervention wie vorhergesagt?

Von diesen vier möglichen Lernaufgaben hat sich die Projektgruppe auf eine Lernaufgabe konzentriert, die eine Mischung aus Alarmfunktion und Interventionsvorhersage ist. Im ersten Semester sollte auf Grund des zeitlichen Verlaufs ausgewählter Variablen (größtenteils Vitalwerte) vorhergesagt werden, wann einem Patienten ein Bolusmedikament verabreicht werden soll. Da diese Data Mining-Experimente nicht sehr erfolgreich waren — es besteht kein kausaler Zusammenhang zwischen den ausgewählten Variablen und den Bolusmedikamenten — wurde die Lernaufgabe im zweiten Semester nach Rücksprache mit Dr. Imhoff dahingehend verändert, dass auf Grund des zeitlichen Verlaufs ausgewählter Variablen die Dosierungsänderungen bestimmter Dauerinfusionen vorhergesagt werden sollen. Diese Data Mining-Versuche orientierten sich an den in [5] beschriebenen Experimenten. Sie wurden u.a. noch dahingehend ausgebaut, dass zusätzlich zur Support Vector Machine (SVM) [9, 8] die Lernverfahren C4.5 [7] und RDT aus der Werkbank MOBAL [6] verwendet wurden.

[2] *Bolusgaben* sind Medikamente, die einem Patienten nur zu einem bestimmten Zeitpunkt verabreicht werden, im Gegensatz zu Dauerinfusionen.

3 Data Mining-Baukasten

3.1 Die Komponenten des Data Mining-Baukastens

Wie schon erwäht hat die Projektgruppe zur Durchführung der Data Mining-Versuche drei Lernverfahren ausgewählt: C4.5 als ein Verfahren zum Erlernen von Entscheidungsbäumen/Regeln, *mySVM* als Implementation einer SVM, und das Regellernverfahren RDT. Direkt auf den rohen Patientendaten kann keines dieser Lernverfahren arbeiten, ganz davon abgesehen, dass das auch nicht sinnvoll ist. Die Daten müssen für die drei Lernverfahren aufbereitet werden, wobei einige Vorverarbeitungsschritte für alle Lernverfahren gleich sind und andere wiederum speziell für ein Lernverfahren benötigt werden. Die einzelnen Vorverarbeitungsschritte sind im folgenden kurz zusammengefasst, Details stehen in Kapitel 4 des Endberichts [3] der Projektgruppe.

Auswahl der Daten: Aus den mehr als 400 Patienten werden einige zum Lernen und andere zum Testen der gelernten Hypothesen ausgewählt. Außerdem wird die Menge der Variablen, die zum Lernen verwendet werden, eingeschränkt.

Ersetzung fehlender Werte: Alle Variablen weisen fehlende Werte auf. Das hängt nicht nur damit zusammen, dass einige Variablen nicht minütliche gemessen wurden (vgl. Abschnitt 2), sondern auch damit, dass manche Sensoren zu bestimmten Zeitpunkten keine Messwerte geliefert haben weil sie z.B. nicht angeschlossen waren, oder, dass der Patient eine Zeit lang nicht auf der Intensivstation war. Diese fehlenden Werte müssen, soweit möglich, durch sinnvolle Werte ersetzt werden.

Erzeugung der Beispiele: Wie in Abschnitt 2 schon erwähnt bestanden die Lernaufgaben darin, die Gabe eines Bolusmedikaments bzw. die Veränderung der Dosis einer Dauerinfusion auf Grund des zeitlichen Verlaufs bestimmter Variablen vorherzusagen. Ein (Lern)Beispiel für eine Lernaufgabe besteht aus einem Zeitfenster einer bestimmten Länge (1-60min.), an dessen Ende eine Intervention vorgenommen wurde, d.h. ein Zeitfenster endet immer zu einem Zeitpunkt, an dem aus den Daten hervorgeht, dass ein Arzt am Patientenbett war. Auf den Patientendaten müssen

alle Zeitfenster einer bestimmten, festen Länge gefunden werden, sie müssen bzgl. des Zielbegriffs klassifiziert werden und es muss möglich sein, die Patientendaten, die innerhalb bestimmter Zeitfenster liegen, auszuwählen.

Spezielle Vorverarbeitungsschritte: Alle bisher beschriebenen Vorverarbeitungsschritte sind für alle drei Lernverfahren notwendig. Manche Vorverarbeitungsschritte sind allerdings optional, oder nur für bestimmte Lernverfahren nötig. Z.B. müssen die Zeitfenster für C4.5 und *mySVM* in eine Attribut-Werte-Darstellung überführt werden, wohingegen MOBAL/RDT prädikatenlogische Fakten benötigt.

Außerdem enthalten die vielen Messwerte innerhalb eines Zeitfensters Messfehler und Rauschen, weshalb es sinnvoll sein kann, Messwerte innerhalb eines Zeitfenstes *zeitlich* zu aggregieren. Dazu werden die Zeitfenster in mehrere Zeitintervalle unterteilt, die Messwerte innerhalb dieser Zeitintervalle durch statistische Größen (Mittelwert, Modus u.ä.) zusammengefasst und diese dann als Lernattribute zu verwendet.

3.2 Das Zusammenspiel der Komponenten

Vorverarbeitung ist sehr aufwändig, und die Aufbereitung der Daten für jeden Data Mining-Versuch von Hand durchzuführen ist nicht sinnvoll. Deshalb war es von Anfang an ein Ziel der Projektgruppe, jeden Vorverarbeitungsschritt durch ein oder mehrere kleine Programme (*Operatoren*) durchführen zu lassen, die sich zu einer sogenannten Data Mining-Kette kombinieren lassen, die einen vollständigen Data Mining-Versuch darstellt, d.h. alle Schritte von den Rohdaten über die gesamte Vorverarbeitung, dem Lernen mit einem der Lernverfahren bis hin zum Testen der gelernten Hypothesen. Eine Data Mining-Kette wird von einem Interpreter ausgeführt, der die Operatoren in der richtigen Reihenfolge mit den richtigen Daten und Parametern aufruft.

Um die Operatoren zu einer Data Mining-Kette integrieren zu können, müssen diese sich an eine gemeinsame Schnittstelle halten, damit der Interpreter, der die Operatoren einer Data Mining-Kette aufrufen soll, nicht die speziellen Aufrufkonventionen jedes Operators kennen muss und der Baukasten erweiterbar bleibt.

Der Data Mining-Baukasten, d.h. die Operatoren zur Vorverarbeitung und zum Aufruf der Lernverfahren sowie der Interpreter, sind in Java programmiert. Die Wahl von Java hatte verschiedene Gründe:

- Die Programmierkenntnisse der einzelnen Projektgruppenteilnehmer waren sehr unterschiedlich, aber da jeder im Grundstudium die objektorientierte Programmiersprache Beta erlernt hat, fiel die Wahl der Programmiersprache auf Java, da Java ebenfalls objekt-orientiert und recht einfach erlernbar ist.
- Java bietet über JDBC[3] die Möglichkeit, auf relationale Datenbanken zuzugreifen.

  Auf Grund der großen Datenmengen, mit denen die Projekgruppe arbeiten musste, wurde entschieden, eine relationale Datenbank zur Verwaltung der Daten zu verwenden. Dazu wurden die Rohdaten in eine Datenbank überspielt, und der weitere Zugriff und die Verarbeitung der Daten wurden mit Hilfe von SQL[4] über die Datenbank abgewickelt. Als Datenbanksystem stand der Projektgruppe ein Oracle-Datenbankserver zur Verfügung, allerdings wurde bei der Implementierung der Operatoren darauf geachtet, dass die verwendeten SQL-Befehle dem SQL-92-Standard entsprechen. Somit sollte der Data Mining-Baukasten mit jedem Datenbanksystem, das über einen JDBC-Treiber verfügt, zusammenarbeiten.
- Java ist plattformunabhängig und bietet die Möglichkeit, während eines Programmlaufs Klassen nachzuladen. Das wird zur Erweiterung des Data Mining-Baukasten benötigt (s.u.).

Um nun die schon erwähnte einheitliche Schnittstelle für die Operatoren sicherzustellen, wurde eine `Operator`-Klasse als Basisklasse für alle Operatoren, die in den Data Mining-Baukasten eingebunden werden sollen, definiert. Diese Basisklasse stellt Methoden zur Verfügung, mit denen einem Operator mitgeteilt werden kann, welche Datenbanktabelle ihm als Eingabe dient und wie die Ausgabetabelle heißen soll; außerdem gibt es eine Methode, mit dem weitere Parameter eines Operators gesetzt werden können, und es gibt Methoden, die die Ausführung eines Operators steuern.

Zur Darstellung der Data Mining-Ketten wird XML [2] verwendet. Dazu wurde eine XML-DTD erstellt, mit der der Datenfluss in einer Data Mining-Kette dargestellt werden kann. In einem XML-Dokument, das eine Data Mining-Kette repräsentiert, ist jeder Operator, der in dieser Data Mining-Kette verwendet wird, aufgeführt. Zu jedem Operator werden die Namen seiner Eingabe- und Ausgabetabelle mitabgespeichert sowie weitere Parameter, die die Arbeitsweise des Operators steuern. Abbildung 1 zeigt den entsprechenden Ausschnitt aus der XML-DTD.

[3] *J*ava *D*ata*b*ase *C*onnectivity
[4] Structured *Q*uery *L*anguage

```
<!ELEMENT operatorinkette (einstellung*)>
<!ATTLIST operatorinkette
          eingabeort     CDATA  #REQUIRED
          ausgabeort     CDATA  #REQUIRED
          operatorname   CDATA  #REQUIRED>

<!ELEMENT einstellung  (notwendigespalte*)>
<!ATTLIST einstellung
          parametername  CDATA  #REQUIRED
          parameterwert  CDATA  #IMPLIED>

<!ELEMENT notwendigespalte (datentyp)+>
<!ATTLIST notwendigespalte
          spaltenname    CDATA  #IMPLIED
          tabellenname   CDATA  #IMPLIED>

<!ELEMENT datentyp
          (sqldatentyp*, metadatentyp+)>
<!ELEMENT sqldatentyp    EMPTY>
<!ATTLIST sqldatentyp
          typ (BINARY | BIT | BLOB | CHAR
              | CLOB | DATE | DECIMAL
              | DOUBLE | FLOAT | INTEGER
              | LONGVARBINARY | LONGVARCHAR
              | NUMERIC | REAL | SMALLINT
              | TIME | TIMESTAMP | TINYINT
              | VARBINARY | VARCHAR
              | VARCHAR2)       #REQUIRED
          groesse        CDATA  #IMPLIED>

<!ELEMENT metadatentyp   EMPTY>
<!ATTLIST metadatentyp
          metadatenflag  CDATA  #REQUIRED>
```

Abbildung 1. Ausschnitt aus der XML-DTD zur Darstellung der Data Mining-Ketten.

`operatorinkette` instanziiert den über das Attribut `operatorname` angegebenen Operator. Die Attribute `eingabeort` und `ausgabeort` enthalten die Namen der Ein- und Ausgabetabelle des Operators, und über das Element `einstellung` können die Parameter des Operators festgelegt werden. Erklärungsbedürftig sind noch restlichen Elemente.

Jeder Operator des Data Mining-Baukastens kann nur mit bestimmten Daten arbeiten, d.h. legt den Tabellen, die er als Eingabe akzeptiert, bestimmte Beschränkungen auf. Der Operator zu Ersetzung fehlender Werte zum Beispiel kann bestimmte Verfahren nur auf numerische Attribute anwenden, und er benötigt in seiner Eingabetabelle auf jeden Fall eine Spalte, die den Zeitindex enthält, sowie eine Spalte mit einer Patienten-ID, da es sich — wie schon mehrmals erwähnt — bei den vorliegenden Patientendaten um Zeitreihen handelt. Andere Operatoren, z.B. solche, die eines der Lernverfahren aufrufen, benötigen in der Eingabe auf jeden Fall eine Spalte, die die Klassifikation eines Beispiels bzgl. des Zielbegriffs enthält.

Solche Einschränkungen und Besonderheiten der Datentabellen werden mit Hilfe von Metadaten, also Daten über Daten, modelliert. Jede Spalte einer Tabelle hat einen SQL-Datentyp, der aussagt, ob die Spalte Zahlen, alphanumerische Werte o.ä. enthält. In der XML-DTD wird der SQL-Datentyp einer von einem Operator auf jeden Falls benötigten Spalte (dargestellt durch `notwendigespalte`) mit dem Element `sqldatentyp` modelliert. Der SQL-Datentyp einer Spalte liefert aber nicht genügend Information. Eine numerische Spalte, die nur ganze Zahlen aufnehmen kann, könnte einen Zeitindex enthalten, eine Patienten-ID oder aber auch ein numerisch kodiertes Attribut. Diese zusätzlichen Informationen werden mit Metadatentypen dargestellt. Ein Zeitindex wäre z.B. durch den Metadatentyp `ZEITINDEX` gekennzeichnet, wohingegen eine Spalte mit einer Patienten-ID den Metadatentyp `PATIENTEN_ID` hat. Metadatentypen sind einfache Zeichenketten, was aber für die Erfordernisse der Projektgruppe vollkommen ausreicht.

So kann z.B. die Einschränkung, dass eine Spalte der Eingabetabelle einen ganzzahligen, 10-stelligen Zeitindex enthält, folgendermaßen dargestellt werden:

```
<datentyp>
  <sqldatentyp typ="NUMERIC"
               groesse="10:0">
  </sqldatentyp>
  <metadatentyp
    metadatenflag="ZEITINDEX">
  </metadatentyp>
</datentyp>
```

Die Metadaten unterstützen den Interpreter bei der Ausführung der Data Mining-Ketten. Wenn der Interpreter eine Kette ausführt, so weiß er genau, welche besonderen Spalten ein Operator in seiner Eingabetabelle benötigt. In einer Verwaltungstabelle werden für jede Spalte jeder Tabelle, die ein Operator anlegt, die Metadaten abgespeichert. Mit Hilfe dieser Tabelle kann der Interpreter die Namen der Spalten einer Eingabetabelle finden, die den von einem Operator benötigten Spalten entsprechen, und dem Operator diese Namen mitteilen. Passt eine Eingabetabelle nicht zu einem Operator, weil von diesem benötigte Spalten fehlen, so meldet der Interpreter einen Fehler. Damit dieses Zusammenspiel zwischen den Operatoren und dem Interpreter funktioniert, müssen die Operatoren natürlich die Metadaten für ihre Ausgabetabelle korrekt setzen. Dafür stellt eine besondere Klasse einige Methoden zur Verfügung, mit denen ein Operator eine Tabelle in der zentralen Verwaltung anmelden kann und die Metadaten der einzelnen Spalten registrieren lassen kann.

Eine weitere Verwendung findet der Metadatenansatz beim Einbinden neuer Operatoren in den Data Mining-Baukasten. Neue Operatoren können über eine Java-Klasse in den Data Mining-Baukasten eingebunden werden. Diese Java-Klasse muss von der `Operator`-

Basisklasse abgeleitet sein, damit der Interpreter diesen Operator aufrufen kann. Das eigentliche Einbinden erfolgt dann über eine XML-Datei. Diese enthält, neben dem Klassennamen des Operators (der benötigt wird, damit die Java-Klasse geladen werden kann) und einer kurzen Beschreibung wie die XML-Datei einer Data Mining-Kette eine Aufzählung der vom neuen Operator für seine Eingabetabelle benötigten Spalten.

3.3 Ein Beispiel für eine Data Mining-Kette

Nach der Erläuterung des Data Mining-Baukastens und des Metadatenansatzes soll jetzt kurz vorgestellt werden, wie ein Data Mining-Baukasten durch Wiederverwendung von Vorverarbeitungsketten und einfachen Austausch von Operatoren die Durchführung von Data Mining-Experimenten erleichtert.

In Abbildung 2 ist eine Data Mining-Kette mit *mySVM* als verwendetem Lernverfahren dargestellt. Die Kästchen mit aufrechter Schrift stellen die einzelnen Operatoren dar, die Kästchen mit schräggestellter Schrift Parametertabellen für die Operatoren und die Pfeile modellieren den Datenfluss.

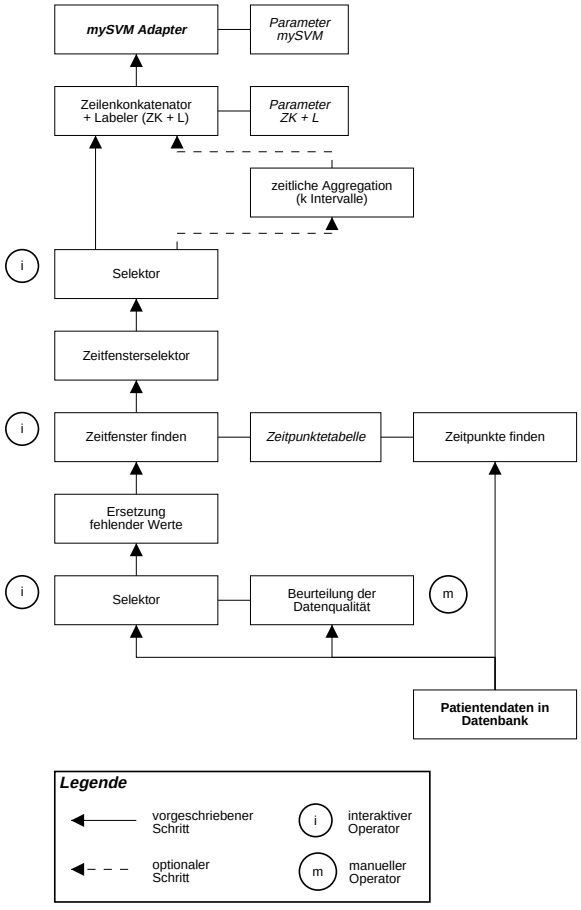

Abbildung 2. Eine Data Mining-Kette für *mySVM*.

Aus den rohen Patientendaten in der Datenbank werden mit dem „Selektor“-Operator die Daten einiger Patienten zum Lernen ausgewählt. Zusätzlich werden die Variablen, die zum Lernen verwendet werden, eingeschränkt, und zwar werden nur solche Variablen ausgewählt, die bei der „Beurteilung der Datenqualität“ als vielversprechend bewertet wurden. Auf dieser Vorauswahl an Daten werden mit dem „Ersetzung fehlender Werte“-Operator fehlende Werte ergänzt und kurze Lücken aufgefüllt.

Die Beispiele werden mit dem „Zeitfenster finden“-Operator erzeugt. Dieser benötigt die vom „Zeitpunkte finden“-Operator zuvor generierte Zeitpunktetabelle, die für alle Patienten alle Zeitpunkte einer Intervention enthält. Da der „Zeitfenster finden“-Operator Zeitfenster unterschiedlicher Länge erzeugt, *mySVM* aber mit einer Attribut-Werte-Darstellung arbeitet und deshalb alle Beispiele, sprich Zeitfenster, die gleiche Größe haben müssen, wählt der „Zeitfensterselektor“-Operator noch Zeitfenster einer bestimmten Länge aus.

Mit einem weiteren „Selektor“-Operator kann die Variablenauswahl noch weiter eingeschränkt werden. Das ist z.B. nützlich, wenn man die zum Lernen am besten geeignete Variablenauswahl finden will. Dann kann man die Ausgabetabelle des „Zeitfensterselektor“-Operators speichern und in einer kleineren Data Mining-Kette, die mit dem zweiten „Selektor“-Operator beginnt, mit verschiedenen Variablenauswahlen experimentieren.

Die Verwendung des „zeitliche Aggregation“-Operators ist optional. Er würde die Zeitfenster nochmals in k Zeitintervalle unterteilen und für diese Zeitintervalle statistische Größen berechnen, die dann als Lernattribute verwendet würden (vgl. Abschnitt 3.1). Der „Zeilenkonkatenator + Labeler“-Operator bereitet die Daten noch weiter auf, bevor der „*mySVM* Adapter“-Operator schließlich das Lernverfahren aufruft.

Die Beschreibung der in Abb. 2 dargestellten Data Mining-Kette ist ziemlich knapp (Details stehen in [3], Kapitel 4), es soll hier aber nur verdeutlicht werden, wie ein Data Mining-Baukasten die Wiederverwendung von Vorverarbeitungsketten unterstützt.

In Abbildung 3 ist eine Data Mining-Kette mit C4.5 als verwendetem Lernverfahren dargestellt. Bis zum zweiten „Selektor“-Operator ist die Vorverarbeitung in beiden Data Mining-Ketten identisch. Der Unterschied besteht nur darin, dass in der C4.5-Kette die Daten dann noch mit dem „Wertediskretisierung“-Operator diskretisiert und/oder zeitlich aggregiert werden können. Der „Wertediskretisierung“-Operator teilt numerische Attribute in Intervalle ein und gibt diesen Intervallen symbolische Namen. Das

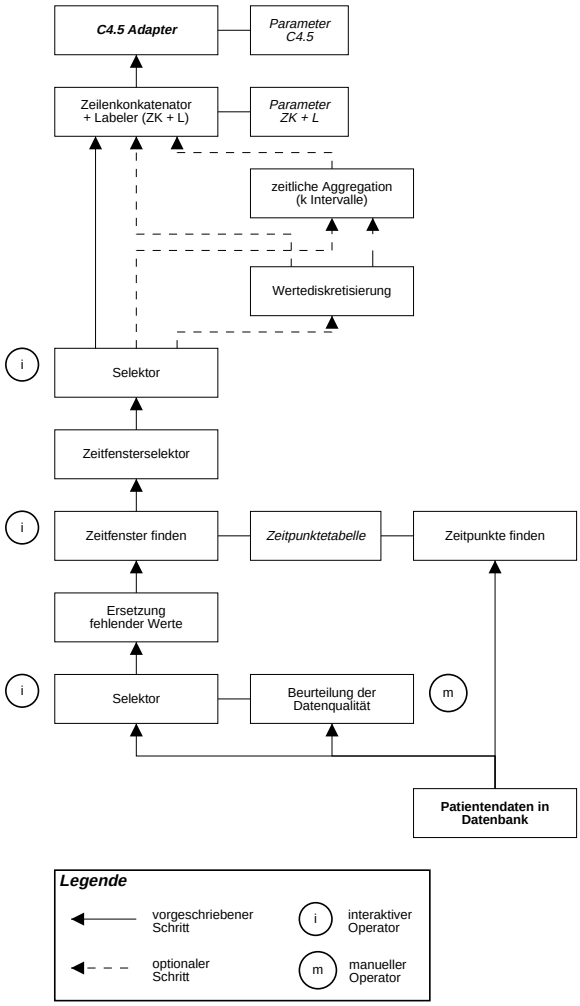

Abbildung 3. Eine Data Mining-Kette für C4.5.

kann sinnvoll sein, um die Verständlichkeit der von C4.5 gelernten Entscheidungsbäume und Regeln zu verbessern.

3.4 Vorteile eines Data Mining-Baukastens

Nachdem bisher die Struktur des Data Mining-Baukastens und seine Verwendung beschrieben wurden, sollen in diesem Unterabschnitt die Vorteile bei Verwendung eines Data Mining-Baukastens dargestellt werden. Dazu wird die Durchführung der beiden in Abschnitt 3.3 vorgestellten Data Mining-Ketten *ohne* Verwendung eines Baukastens beschrieben.

Natürlich sind Data Mining-Ketten ähnlich der in Abschnitt 3.3 dargestellten auch ohne Verwendung eines Baukastens möglich. Die gesamte Vorverarbeitung der Daten und der Aufruf des Lernverfahrens müssen dann von Hand durchgeführt oder anderweitig als mit einem Interpreter, z.B. mit einem Shell-Skript, automatisiert werden.

Viele der Operatoren erledigen Aufgaben, die mit einigen SQL-Befehlen recht einfach erledigt werden können. Der „Selektor“-Operator zum Beispiel erzeugt eine View seiner Eingabetabelle, die bestimmten Kriterien genügt, d.h. nur bestimmte Spalten der Eingabetabelle und nur die Daten bestimmter Patienten enthält. So etwas kann man mit einem einzigen SQL-Befehl durchführen.

Andere Operatoren hingegen erfüllen komplexe Aufgaben, die nicht direkt in SQL durchgeführt werden können. Beispiele sind der „Ersetzung fehlender Werte“-Operator und der „Zeilenkonkatenator + Labeler“-Operator. Wollte man diese Vorverarbeitungsschritte von Hand durchführen, so müsste man auf eine Programmiersprache zurückgreifen oder eine prozedurale Erweiterung von SQL verwenden.

Ein entscheidenderer Vorteil als die Kapselung komplexer Aufgaben ist die Wiederverwendbarkeit sowohl der Operatoren als auch der Data Mining-(Teil)Ketten. Wie in Abschnitt 3.2 beschrieben, müssen sich alle Operatoren, die in den Data Mining-Baukasten integriert werden sollen, an eine bestimmte Aufrufschnittstelle halten, damit der Interpreter die Data Mining-Ketten ausführen kann. Dadurch können die Operatoren für ganz andere Data Mining-Versuche verwendet werden, als nur für die, für die sie ursprünglich implementiert wurden. Man muss nur eine entsprechende Data Mining-Kette zusammenbauen und diese vom Interpreter ausführen lassen.

Diese Wiederverwendung funktioniert natürlich umso besser, je allgemeiner die Aufgabe ist, die ein Operator erledigt. Ein Operator wie der „Zeilenkonkatenator + Labeler“ ist ziemlich spezifisch auf die von der Projektgruppe durchgeführten Versuche zugeschnitten und somit nur bedingt in anderen Experimenten einsetzbar. Aber der „Wertediskretisierung“-Operator, der numerische Werte zu Intervallen zusammenfasst, ist in vielen Data Mining-Ketten einsetzbar, da die Diskretisierung von numerischen Daten eine recht allgemeine Aufgabe ist. Mit ein wenig Arbeit wären auch die Operatoren „Ersetzung fehlender Werte“ und „zeitliche Aggregation“ allgemeiner einsetzbar. Man müsste nur die Verwendung von ganzen Zahlen als Zeitstempel auf einen SQL-Datentyp wie z.B. `DATE` oder `TIMESTAMP` umstellen.

Man kann aber nicht nur einzelne Operatoren, sondern auch ganze Data Mining-Ketten oder Teile davon wiederverwenden, wie im vorherigen Abschnitt schon gezeigt wurde. Dadurch kann man z.B. eine Bibliothek an Vorverarbeitungsketten anlegen und bei neuen Data Mining-Versuchen wiederverwenden, oder auch komplexere Ketten aus einfacheren zusammenbauen. Soetwas kann auch neuen Benutzern bei der Verwendung

des Baukastens helfen. Sie können eine funktionierende Data Mining-Kette nehmen und mit ihr experimentieren, indem sie z.B. Parameter ändern oder Operatoren entfernen oder hinzufügen. Dabei werden sie von den Metadaten unterstützt.

Wie schon erwähnt, beschreiben die Metadaten den Inhalt einzelner Tabellenspalten, und zusammen beschreiben die Metadaten einzelner Tabellenspalten die Daten einer Tabelle. So kann man, selbst ohne die einzelnen Vorverarbeitungsschritte zu kennen, aus den Metadaten zu einer Tabelle eine Beschreibung der Daten in dieser Tabelle erhalten. Ohne einen Data Mining-Baukasten mit integriertem Metadatenansatz müsste man von Hand über die erstellten Tabllen und die in ihnen enthaltenen Daten buchführen. Mit Hilfe der Metadaten gschieht das weitestgehend automatisch, und man weiß jederzeit, was für Daten in einer Datenbank gespeichert sind.

Die einzige Situation, in der man dem System von Hand die Metadaten mitteilen muss, ist bei der Einspielung neuer Rohdaten. Danach funktionieren die bereits erstellten Data Mining-Ketten aber auch mit den neuen Daten, sofern die Daten mit den Ketten überhaupt verarbeitet werden können. Das klappt sogar dann, wenn die Daten zwar das richtige Format haben, die einzelnen Spalten der neuen Tabelle aber ganz andere Namen haben als bisher. Mit Hilfe der Metadaten „finden“ die Operatoren die von ihnen auf jeden Fall benötigten Spalten, vgl. Abbschnitt 3.2. Ohne die Metadaten müssten die Namen der unbedingt notwendigen Spalten von Hand in die Data Mining-Kette eingetragen werden, was die Wartbarkeit erheblich beeinträchtigt.

Ein weiterer, nicht zu unterschätzender Vorteil eines Data Mining-Baukastens ist, dass man Data Mining-Experimente bei entsprechender Unterstützung durch den Interpreter parallel laufen lassen kann. Der Projektgruppe stand für ihre Experimente ein Rechner-Pool zur Verfügung. Dadurch konnten auf jedem Rechner Data Mining-Versuche durchgeführt werden. Da alle Rechner auf dieselbe Datenbank zugriffen, musste der Interpreter dafür Sorge tragen, dass nicht versehentlich noch benötigte Daten überschrieben wurden, und dass nicht mehr benötigte Tabellen gelöscht wurden um Speicherplatz zu sparen. Von Hand wäre soetwas nur sehr schwer zu koordinieren. Wegen der zentralen Verwaltung der Tabellen durch den Interpreter war das aber recht einfach zu automatisieren.

Ein Vergleich des von der Projektgruppe implementierten Data Mining-Baukastens mit kommerziellen Systemen wäre wünschenswert, ist aber leider nicht machbar, weil der Autor keinen Zugang zu solchen Systemen hat.

4 Diskussion

In diesem Artikel wurde die Arbeit der Projektgruppe MEDMINER vorgestellt. Aufgabe der Projektgruppe war das Lernen auf intensivmedizinischen Daten und das Entwickeln eines Data Mining-Baukastens. Es wurde gezeigt, dass ein Data Mining-Baukasten, kombiniert mit einem Metadatenansatz, die Durchführung von Data Mining-Versuchen erleichtert, da die Vorverarbeitung in kleine Schritte aufgeteilt ist, die von Operatoren durchgeführt werden, die nahezu beliebig miteinander kombiniert werden können. Die Verwendung von Java als Programmiersprache und der Metadatenansatz ermöglichen außerdem eine Erweiterung des Baukastens um weitere Vorverarbeitungsoperatoren oder Lernverfahren.

Der verwendete Metadatenansatz und die Beschreibung der Data Mining-Ketten mit Hilfe einer XML-DTD reichen für die Belange der Projektgruppe vollkommen aus. Für umfangreichere Data Mining-Baukästen wäre es aber denkbar, einen mächtigeren Metadatenansatz, basierend z.B. auf dem Common Warehouse Metamodel (CWM) [1], zu entwerfen, mit dem man die Transformation der Daten, die die Operatoren vornehmen, besser darstellen kann. Aufbauend darauf könnte man so etwas wie Metalernen betreiben, d.h. den Benutzer des Systems beim Zusammenstellen von neuen Data Mining-Ketten unterstützen, indem, ausgehend von bekannten, für bestimmte Daten gut geeigneten Data Mining-Ketten, das System den Benutzer berät, welcher weitere Vorverarbeitungsschritt bei den vorliegenden Daten vielversprechend ist. Das CWM bietet auch die Möglichkeit, die gelernten Modelle und Hypothesen darzustellen; etwas, was der von der Projektgruppe entworfene Formalismus nicht kann, was aber auch nicht vorgesehen war.

Den Formalismus könnte man dahingehend erweitern, dass Data Mining-Ketten geschachtelt werden können. Dann könnte man eine komplette Data Mining-Kette wie einen Operator in eine weitere Data Mining-Kette integrieren, um z.B. mit dieser Kette eine Kreuzvalidierung durchzuführen. Man könnte den Formalismus also um einige programmiersprachliche Konstrukte erweitern. Ebenso könnte man eine graphische Benutzeroberfläche erstellen, um den Benutzer beim Erstellen der Data Mining-Ketten zu unterstützen und ihn z.B. zu warnen, falls er zwei nicht kompatible Operatoren kombinieren möchte.

Danksagung. Die in diesem Artikel vorgestellten Forschungsergebnisse sind das Resultat der Zusammenarbeit aller neben mir an der Projektgruppe MEDMINER beteiligten Studenten: Twain Hoffmann, Valeri Martchouk, Michael Parohl, Sandra Pires Costa, Holger Smura, Sandra Solbach, Oliver Trendelkamp, Selvinaz Vurankaya und Markus Wagner. Bei ihnen möchte

ich mich bedanken, ebenso bei unseren beiden Betreuern, Ralf Klinkenberg und Stefan Rüping.

Literatur

1. D.T Chang, S. Iyengar et al.: *Common Warehouse Metamodel (CWM) Specification*. Technischer Bericht OMG, 2000. Online verfügbar unter `http://www.omg.org/`.
2. R. Eckstein: *XML kurz & gut*. O'Reilly Verlag, Köln, 2000.
3. J. Fisseler, T. Hoffmann, V. Martchouk, M. Parohl, S. Pires Costa, H. Smura, S. Solbach, O. Trendelkamp, S. Vurankaya und M. Wagner: MEDMINER — *Maschinelles Lernen in der Intensivmedizin*. Technischer Bericht Universität Dortmund, Fachbereich Informatik, 2001.
4. M. Imhoff: *Intensivmedizinische Verlaufsdaten*. Technischer Bericht Städtische Kliniken Dortmund, Oktober 2000.
5. K. Morik, M. Imhoff, P. Brockhausen, T. Joachims und U. Gather: *Knowledge Discovery and Knowledge Validation in Intensive Care*. Artificial Intelligence in Medicine, 19:225–249, 2000.
6. K. Morik, S. Wrobel, J.-U. Kietz und W. Emde: *Knowledge Acquisition and Machine Learning: Theory, Methods, and Applications*. Academic Press, London, 1993.
7. J. R. Quinlan: *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, California, 1993.
8. B. Schölkopf, C. J. C. Burges und A. J. Smola: *Advances in Kernel Methods — Support Vector Learning*. MIT Press, Cambridge, Massachusetts, 1999.
9. V. Vapnik: *Statistical Learning Theory*. Wiley, 1998.