Ähnlichkeitsbasiertes Retrieval von Fällen durch assoziative Suche in einem mehrdimensionalen Datenraum

Hannes Öchsner, Stefan Wess
Universität Kaiserslautern, Fachbereich Informatik
AG Künstliche Intelligenz - Expertensysteme
Prof. Dr. M.M. Richter

1 Einleitung

Ein fundamentaler Schritt des fallbasierten Schließens (FBS) ist das Retrieval einer handhabbaren Menge von Fällen aus einer Fallbasis, die als Grundlage für die weiteren Schritte des FBS, wie die Modifikation und Übertragung bekannter Lösungen auf einen gegebenen Problemfall, dienen.

Die Relevanz eines Falles kann in dieser Vorauswahl z.B. über die Ähnlichkeit, die seine Repräsentation mit der des Problemfalles hat, bestimmt werden. Ein solches ähnlichkeitsbasiertes Retrieval entspricht der Suche nach den besten Matches für einen gegebenen Fall. Bei umfangreichen Fallbasen stellt sich hierbei vor allem die Forderung nach einem effizienten Retrievalmechanismus, der möglichst nur die in Frage kommenden Fälle auf ihre Ähnlichkeit hin untersuchen und somit ein Durchsuchen der gesamten Fallbasis vermeiden soll.

Hierzu wird nun im folgenden ein Ansatz betrachtet, der versucht, dieses Best-Match-Problem über eine assoziative Suche auf einer mehrdimensionalen Zugriffspfadstruktur für Fallbeschreibungen zu lösen. Der für eine solche Suche benötigte zeitliche Aufwand liegt dabei in der Größenordnung von $O(\log n)$ bezogen auf die Anzahl der Fälle.

Abschnitt 2 liefert dazu eine kurze Übersicht über Annahmen, Strukturen und Algorithmen. In Abschnitt 3 erfolgt eine Erweiterung des Ansatzes um ein Typkonzept mit dem Ziel der Integration verschiedener benutzerdefinierter Wertebereiche mit unterschiedlichen Ähnlichkeitsmaßen.

2 Ähnlichkeitssuche

Das Problem, zu einer vorliegenden Fallbeschreibung die ähnlichsten Fälle (best matches) aus einer Menge gegebener Fälle zu bestimmen, kann man als Suche nach den nächsten Nachbarn (nearest neighbors) auffassen. Bei einer entsprechenden Repräsentation der Fallbeschreibungen können die Fälle als Punkte in einem mehrdimensionalen Datenraum angesehen werden, der dann assoziativ nach den Fällen durchsucht werden muß, deren Beschreibungen zu der des vorliegenden Falles die größten Ähnlichkeiten aufweisen.

Assoziatives Retrieval wird oft auch als Sekundärschlüssel-Suche bezeichnet, wobei jeder Fall als ein geordnetes Tupel von Attributausprägungen in Form von Werten dargestellt ist. Diese Werte repräsentieren die Suchschlüssel eines Falles.

Eine Anfrage an das Retrieval-System, eine sog. *Query*, spezifiziert durch einen solchen Suchschlüssel Bedingungen an die Attributwerte, die die Fallbeschreibungen der Fälle, die als Teil des Anfrageergebnisses zurückgeliefert werden sollen, erfüllen müssen. Im Zusammenhang mit der Auswertung dieser Art von Queries spricht man dann von assoziativer Suche.

In unserer Arbeit gehen wir von Fallbeschreibungen aus, die durch Attribut-Wert-Paare repräsentiert sind, wobei einem Attribut jeweils ein Wertebereich zugeordnet wird. Die Dimensionalität des Datenraumes, der die Fälle als Punkte enthält, ist somit durch die Anzahl der Repräsentationsattribute gegeben, und die Attribute selbst entsprechen den Koordinaten der Punkte.

Zur Unterstützung der Best-Match-Suche werden nun die Fallbeschreibungen in einem sog. k-d-Baum angeordnet. Ein k-d-Baum ist eine in (Be75) vorgeschlagene Erweiterung eines einfachen binären Suchbaumes auf k-dimensionale Suchschlüssel, wobei k größer als 1 ist.

Bentley zeigt, daß diese Datenstruktur die assoziative Suche und die verschiedenen in Retrieval-Systemen auftretenden Anfragetypen unterstützt. Insbesondere zeigt er, daß Best-Match-Queries in Abhängigkeit von der Anzahl der vorhandenen Fälle mit einem durchschnittlichen Aufwand der Größenordnung $O(\log n)$ evaluiert werden können. Voraussetzung dafür ist jedoch ein optimierter Aufbau des

k-d-Baumes.

(FrBeFi77) beschreiben eine Prozedur, die die für den Aufbau durchzuführenden Partitionierungen einer Menge gegebener Punkte auf der Basis von Distanzfunktionen für deren einzelne Koordinaten durchführt. Die Auswahl der Attribute, nach denen jeweils partitioniert wird, richtet sich nach einem Maß für die Streuung der in den Punktkoordinaten auftretenden Werte. Zur Berechnung der Streuung werden die Koordinaten-Distanzfunktionen herangezogen.

Als Eigenschaften werden für diese Koordinaten-Distanzfunktionen Symmetrie und Monotonie gefordert. Eine Funktion, durch die dann die Distanz zwischen zwei Punkten eines k-dimensionalen Datenraumes bestimmt wird, muß darüber hinaus noch zwei weitere Bedingungen erfüllen. Erstens darf die Distanz nicht kleiner werden, wenn in irgendeiner Dimension des Datenraumes der lineare Abstand zwischen den entsprechenden Koordinatenwerten der beiden Punkte größer wird, und zweitens muß die Summe der Distanzen für eine Untermenge der Koordinaten kleiner oder gleich der Gesamtdistanz zwischen den Punkten sein.

Die in (FrBeFi77) vorgenommene Aufwandsbetrachtung für den Aufbau eines optimierten k-d-Baumes bei n gegebenen Punkten ergibt eine Zeitkomplexität der Größenordnung $O(k \cdot n \log n)$. Diesen Aufwand kann man als Kosten einer Preprocessing– oder Lernphase interpretieren, die als Voraussetzung für die Effizienz der anschließenden operationalen Phase des eigentlichen Suchens durchgeführt werden muß.

Es erscheint jedoch nicht unbedingt notwendig, jeweils sofort nach jedem Fall, der neu in die Fallbasis eingebracht wurde, eine Reorganisation der Zugriffspfadstruktur vorzunehmen. Im allgemeinen kann man davon ausgehen, daß auch eine eventuell nur periodisch durchgeführte Reorganisation durchaus noch für eine akzeptable Laufzeit der Suche ausreichend ist. Das Retrieval liefert auf jeden Fall ein richtiges Ergebnis, da die Korrektheit des in Friedman angegebenen Suchalgorithmus von der Optimierung der Struktur unabhängig ist.

In unserer Arbeit wurde der Ansatz nun insofern modifiziert, als für den Aufbau des Baumes *Ähnlichkeitsmaße* Verwendung finden. In diesem Zusammenhang wurde ein Konzept entwickelt, das die Definition wertebereichsspezifischer Ähnlichkeitsmaße erlaubt. Eine kurze Beschreibung des Typkonzeptes erfolgt in Abschnitt 3.

(RiWe91) zeigt, daß eine Transformation von Distanz- in Ähnlichkeitsmaßen durchgeführt werden kann, wenn zwischen beiden Maßen die sog. Kompatibilitätsrelation besteht. Diese Voraussetzung ist bei Erfüllung der oben geforderten Eigenschaften für die Maße gewährleistet. Somit können die bisher gemachten Aussagen in analoger Weise auch auf Ähnlichkeitsmaße übertragen werden.

Die Ähnlichkeit zwischen zwei Fällen A und B wird über eine k-stellige Funktion F bestimmt, wobei k der Dimensionalität des Datenraumes entspricht, so daß gilt

$$similarity(A, B) = F(\mu_1(A_1, B_1), \ldots, \mu_k(A_k, B_k))$$

Ein Fall ist dabei durch einen k-dimensionalen Vektor beschrieben, dessen j-te Komponente einen Wert aus dem Wertebereich D_j des j-ten Repräsentationsattributes enthält. Die Funktion $\mu_j : D_j \times D_j \rightarrow [0, 1]$ bezeichnet das auf diesem Wertebereich definierte Ähnlichkeitsmaß für zwei entsprechende Werte.

Somit gilt für den nächsten Nachbarn N einer gegebenen Fallbeschreibung Q und jeden beliebigen Fall C aus der Fallbasis

$$similarity(Q, N) \geq similarity(Q, C)$$

Dem implementierten Suchalgorithmus, der die Struktur des k-d-Baumes ausnutzt, liegt die Idee der inkrementellen Version aus (Br90) zugrunde. Diese Version ermöglicht es unter Verwendung der grundlegenden Ideen aus (FrBeFi77), effizient eine beliebige Anzahl von nächsten Nachbarn inkrementell zu bestimmen, ohne daß diese Anzahl im voraus festgelegt werden muß.

Als eine zusätzliche Erweiterung können für die Suche beliebige „k.o.-Kriterien“ als Constraints spezifiziert werden, die jeder der ähnlichsten Fälle unbedingt erfüllen muß. Ist dies bei einem untersuchten Fall nicht gegeben, so qualifiziert sich der Fall auch bei einer noch so großen Ähnlichkeit nicht als Kandidat für die ähnlichsten Fälle.

3 Ein Typkonzept

Damit die beschriebenen Suchstrukturen und –algorithmen auch in Anwendungen benutzt werden können, in denen Fälle nicht nur durch numerische Daten repräsentiert sind, ist in unserer Implementierung ein Typkonzept integriert, das es erlaubt, verschiedene anwendungsabhängige Wertebereiche zu definieren, die in gewisser Weise benutzerdefinierten Datentypen entsprechen.

Diese Typen werden als Instanzen einer allgemeinen Typbeschreibungsklasse spezifiziert. Die Instanzen enthalten den ihnen zugrundeliegenden geordneten Wertebereich und ein darauf definiertes Ähnlichkeitsmaß. Ihre Ordnungen und Ähnlichkeitsmaße werden zum Aufbau der k-d-Bäume und zur assoziativen Suche herangezogen. Sie können jeweils für je zwei Werte des entsprechenden Wertebereichs entweder in Form einer 2-dimensionalen Matrix oder als zweistellige Funktion angegeben werden.

Desweiteren ist es möglich, durch die Verknüpfung von Typen über die *is-a*-Relation ganze Typhierarchien aufzubauen, in denen in Anlehnung an das Paradigma der Objektorientierung Wertebereiche und Ähnlichkeitsmaße übergeordneter Typen (Supertypen) auf deren Subtypen vererbt bzw. dort überschrieben werden können.

Wir implementierten dieses Konzept zur Definition anwendungsspezifischer Datentypen als ein generisches Modul, das die einfachen Basistypen *Number*, *Float*, *Integer*, *String*, *Symbol* und *Boolean* (sowohl mit symmetrischem als auch mit asymmetrischem Ähnlichkeitsmaß) bereitstellt. Für jeden Basistyp ist ein Default-Ähnlichkeitsmaß vorgegeben, bei den drei numerischen Basistypen z.B. das euklidische Maß.

Die Wertebereiche der Typen müssen geordnete Mengen sein. Sie können in Form von Listen bzw. Intervallen, die durch die Anordnung ihrer Elemente implizit schon eine Ordnung auf den Elementen definieren, angegeben werden oder als Mengen, für deren Elemente dann zusätzlich eine Ordnungsrelation explizit definiert werden muß. Die Elemente der Wertebereiche sind entweder einfache Grundtypen oder selbst wieder solche Mengen.

Die allgemeine Typbeschreibungsklasse wurde als OPAL–Klasse (Serv86) implementiert, und jeder benutzerdefinierte Datentyp entspricht einer Instanz dieser Klasse. Zur Veranschaulichung sind im folgenden zwei Typdefinitionen aufgeführt, zum einen der Basistyp *Number* und zum anderen exemplarisch ein benutzerdefinierter Typ *MyNumbers*. Die Syntax zur Darstellung der Attributwerte der Instanzen orientiert sich grob an OPAL.

Beispiel:

Number

```
name       : 'Number'
supertype  : nil
similarity : [:val1 :val2 |
               1/(1+(val1−val2)abs)]
range      : nil
valueType  : Number
```

MyNumbers

```
name       : 'MyNumbers'
supertype  : 'Number'
similarity : nil
range      : Interval
                 withElements : #[5,15]
                 constrainedTo: Integer
valueType  : Integer
```

Die *is-a*-Relation zwischen zwei Typen ist durch das Attribut `supertype` repräsentiert. Der Wert dieses Attributes enthält die Referenz zu einem übergeordneten Datentyp. Im Beispiel ist der Basistyp *Number* der Supertyp zu *MyNumbers*, während *Number* selbst keine Relation zu einem ihm übergeordneten Typ mehr hat, was durch den `nil`-Wert zum Ausdruck kommt. Das Attribut `valueType` gibt die Smalltalk-Klasse an, aus deren Instanzen sich der Wertebereich (`range`) und somit auch der Ergebniswert des benutzerdefinierten Datentyps rekrutiert. Ein `nil`-Wert in `range` heißt, daß darüber hinaus keine Wertebereichseinschränkung besteht. Bei Datentypen, für die ein Supertyp spezifiziert ist, bedeuten `nil`-Werte in den Attributen `similarity`, `range` und `valueType` noch zusätzlich, daß hier jeweils automatisch die entsprechenden Werte des Supertyps übernommen werden sollen (Prinzip der Vererbung).

Beispielsweise umfaßt der Wertebereich für *Number* alle Instanzen der Smalltalk-Klasse `Number`, während der Bereich von *MyNumbers* auf `Integer`-Instanzen zwischen `5` und `15` eingeschränkt ist. Das für *Number* in Form eines 2-Variablen-Blockes angegebene "euklidische Ähnlichkeitsmaß„ wird ebenfalls auf *MyNumbers* weitervererbt.

Auf diese Art lassen sich ganze Hierarchien der verschiedensten Datentypen mit nahezu beliebig komplexen Wertebereichen aufbauen.

Durch das Konzept wird somit ein Mechanismus bereitgestellt, der das Retrieval von Fällen mit Hilfe der k-d-Bäume auch für anwendungsspezifische Datentypen und entsprechende Ähnlichkeitsmaße ermöglicht.

4 Implementierung

Eine Implementierung (vgl. (Öchsner92)) der hier beschriebenen Ideen erfolgte auf dem objektorientierten Datenbanksystem GemStone (MaStOtPu86) in der dort zur Verfügung stehenden Datenbanksprache OPAL (Serv86). Das GemStone DBMS bietet eine Anwenderschnittstelle zu Smalltalk-80 (GoRo83). Die Fallbasis wurde in Form einer GemStone-Datenbank realisiert, in der Fälle persistent gehalten werden können. Zur Unterstützung des ähnlichkeitsbasierten Retrievals können sowohl anwendungsabhängige Datentypen als auch die oben beschriebene Zugriffspfadstruktur des k-d-Baums für verschiedene Suchschlüssel durch den Benutzer erzeugt werden.

Die Fallbasis und die benötigten Strukturen sind an sich anwendungsunabhängig. Die einzige Forderung besteht in der Repräsentation der Fallbeschreibungen auf der Basis von Attribut-Wert-Paaren.

Ein weiterer Vorteil ist die Möglichkeit, die für das Retrieval notwendigen DB-Operationen in einem eigenständigen Prozeß unabhängig von der eigentlichen Anwendung ablaufen zu lassen. Dadurch kann die Anwendung auch während der Zeit weiterlaufen, in der die Zugriffe auf die Datenbank erfolgen (Retrieval im Sinne der *any-time* Algorithmen von (RuZi91)).

5 Ausblick

Der vorgestellte Ansatz stellt für eine Menge von Fallbeschreibungen eine mehrdimensionale Zugriffspfadstruktur als Unterstützung des assoziativen Retrievals bereit. Der entscheidende Punkt liegt dabei im Aufbau eines *optimierten* k-d-Baumes, d.h. in der Bestimmung der einzelnen Suchschlüssel, nach denen die Menge der Fälle jeweils partitioniert wird. Diese Auswahl erfolgt hier nach einem statistischen Maß für die Streuung der verschiedenen Attributwerte. In die Berechnung der Streuungsmaßzahlen gehen die auf Attributebene definierten Ähnlichkeitsmaße mit ein.

Man könnte das Problem einer geeigneten Organisation der Fälle in der Fallbasis jedoch auch vom Standpunkt des begrifflichen Gruppierens (conceptual clustering) betrachten, das ein Kernproblem des Maschinellen Lernens darstellt. Ziel wäre es dabei, die Fälle so in konzeptuelle Klassen einzuteilen, daß jede Klasse eine Menge von Fällen enthält, die zueinander „sehr ähnlich" sind, d.h. ähnlicher als zu den Fällen der anderen Klassen. Somit bestünde das Problem des ähnlichkeitsbasierten Retrievals darin, zu einem gegebenen Problemfall dessen Klasse zu bestimmen.

Interessante Ansätze, die in diese Richtung des „conceptual retrieval" gehen, wären beispielsweise Ansätze des assoziativen Retrievals in semantischen Netzwerken, deren Grundlage jedoch gerade in einer konzeptuellen Organisation der Fälle besteht, was insofern ein Problem darstellt, als oft die einzelnen Konzepte von vorneherein nicht bekannt sind. Die oben beschriebene Verwendung der Ähnlichkeitsmaße bei der Partitionierung der Fälle bildet eine nur sehr grobe Annäherung an diese Fragestellung.

Somit wäre im weiteren eventuell noch zu untersuchen, inwiefern bekannte Ideen und Methoden des Maschinellen Lernens gerade im Hinblick auf diesen Punkt eine Integration in den hier vorgestellten Ansatz finden könnten.

Literatur

J.L. Bentley, Multidimensional binary search trees used for associative searching, *Commun ACM* 18, 509-517 (1975).

A.J. Broder, Strategies for efficient incremental nearest neighbor search, *Pattern Recognition* 23, 171-178 (1990).

J.H. Friedman, J.L. Bentley and R.A. Finkel, An algorithm for finding best matches in logarithmic expected time, *ACM Trans. math. Software* 3, 209-226 (1977).

A. Goldberg and D. Robson, *Smalltalk-80: The Language and Its Implementation*, Addison-Wesley, Reading, MA (1983).

D. Maier, J. Stein, A. Otis, and A. Purdy, Development of an object-oriented DBMS, *OOPSLA '86 Proceedings*, ACM, New York, 472-482 (1986).

H. Öchsner, Ähnlichkeitsbasiertes Retrieval von Fällen durch assoziative Suche in einem mehrdimensionalen Datenraum, Diplomarbeit, Universität Kaiserslautern, Fachbereich Informatik, in Vorbereitung (1992)

M.M. Richter and S. Wess, Similarity, Uncertainty and Case-Based Reasoning in PATDEX, in: R.S. Boyer (Hrsg.)*Automated Reasoning*, Kluwer Academic Publisher, 249-265 (1991).

S.J. Russel and S. Zilberstein, Composing real-time systems. In *Proc. IJCAI-91*, Sydney, Australia, 212-217, (1991).

Servio Logic, *Programming in OPAL*, Servio Logic Development Corp., Beaverton, Oregon (1986).