Angewandte Systeme der künstlichen Intelligenz

Kurzübersicht und Einstieg

Walther-Matthias Riedel und Steffen Schoenwiese

September 2004

Inhaltsverzeichnis

1 Quellenangabe

Alle Ansätze, Formeln und Quellen sind **dem** Standardwerk zum Thema KI „Artificial Intelligence - A Modern Approach, Stuart Russel and Peter Norvig, Prentice Hall“ entnommen.

Zum besseren Verständnis wurde jedoch die Aufbereitung und Beispielgebung des teilweise sehr anspruchsvollen Stoffs überarbeitet.

2 Case based Reasoning

2.1 Ausgangssituation

2.1.1 Die Idee

Beim Case-based reasoning geht man implizit von der Annahme aus, dass ähnliche Problemsituationen auch ähnliche Problemlösungen erfordern. Daher wird, um ein neues Problem zu lösen, ein ähnlicher Fall in der vorhandenen Fallsammlung ermittelt und anschließend die vorhandene Problemlösung auf das neue Problem komplett oder in adaptierter Form übertragen. Dadurch ist ein neuer Fall entstanden, der nun in der Fallsammlung aufgenommen werden kann, um bei zukünftigen Problemen ebenfalls herangezogen werden zu können.

2.1.2 Ein Beispiel

> *Autowerkstatt "Car&Bike": es ist kurz vor Feierabend. Automechaniker Stefan H. will gerade abschließen als noch ein letzter Kunde kommt. Das Problem mit seinem Wagen liegt darin, dass der Motor nicht mehr anspringt. Nach einer ersten oberflächlichen Überprüfung stellt er fest, dass sowohl genug Benzin im Tank ist, als auch Öl und Wasser stimmen und keine, auf den ersten Blick wahrnehmbaren, Schäden an Keilriemen oder Lichtmaschine zu erkennen sind. Stefan H. überlegt woran es liegen könnte. Da fällt ihm der Kunde von letztem Monat ein: da hatte der Wagen die gleichen Symptome und nach langer Suche wurde das Übel in Form der kaputtgegangenen Einspritzpumpe identifiziert. Diesmal wird Stefan H. sofort mal nach der Einspritzpumpe sehen...*

2.2 Formale Definitionen

(a) P sei eine Menge von Problemen. $P_i \in P$

(b) L sei eine zugehörige Menge von Lösungen.
$L_i \in L$ sei die Lösung zu $P_i \in P$.

(c) C ist eine Menge von Tupeln.
$(P_i, L_i) \in C$ ist ein Tupel aus Problem und Lösung . Ein konkreter Fall. (P, L) ist ein Tupel aus einem aktuellen Problem und einer noch zu ermittelnder Lösung.

(d) Nun soll L bestimmt werden, indem man nach einem Fall-Tupel $(P_i, L_i) \in C$ sucht, dessen P_i dem konkreten Problem P möglichst ähnlich ist: $P \sim P_i$, in der aufrechten Hoffnung, dass auch $L \sim L_i$ gilt.

2.2.1 Ein Fall

Ein Fall besteht also aus dem Problem-Lösungs-Tupel, welches wir mit einem Namen versehen. Zudem wird dieser Struktur eine Liste mit Attribut-Wert-Paaren zugeornet. Diese Liste spielt eine zentrale Rolle bei der Ähnlichkeitsbewertung verschiedener Fälle.

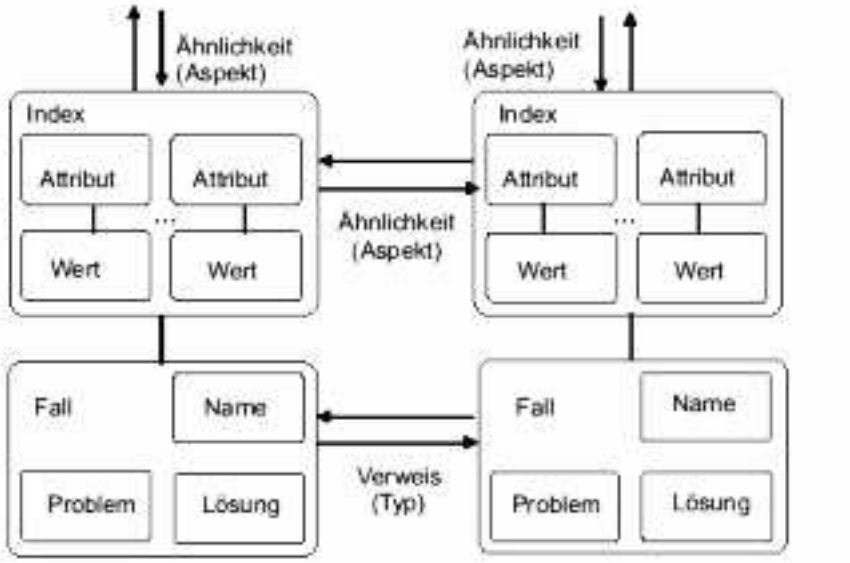

Abbildung 1: Fallrepresentation - allgemein

Ein neues Problem wird nun hinsichtlich seiner Eigenschaften mit denen schon vorhandener Fälle verglichen. Es muss also eine quantitative Aussage über die Ähnlichkeit des neuen Problems mit anderen, schon vorhandenen Fällen der Wissensbasis, gemacht werden. Hierbei tritt es sehr häufig auf, dass Anzahl und Aussage der Attribute nicht identisch sind. Es ist also notwendig eine Schnittmenge zu bilden und so eine Grundlage des Vergleichens zu schaffen.

> *Mann kann sozusagen Äpfel und Birnen nur hinsichtlich bestimmter Aspekte vergleichen. Z.B. Gewicht, Wassergehalt, Vitamin-C-Gehalt u.s.w..*

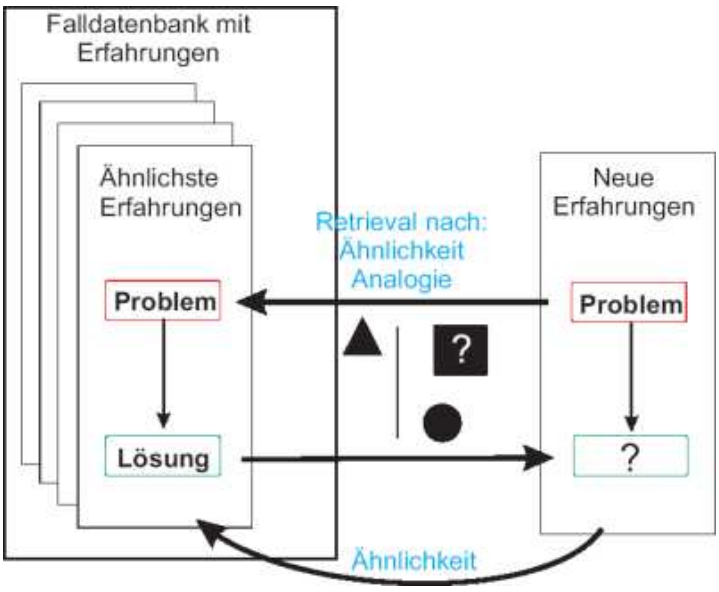

Abbildung 2: Schema Wissensbasis

2.2.2 Der Ähnlichkeitsbegriff

Der Begriff „Ähnlichkeit“ besitzt zwei für uns wesentliche Dimensionen.

(1.) Die inhaltliche Bedeutung

- Ähnlichkeit bezieht sich immer auf einen speziellen Aspekt oder Zweck.
  (Apfel und Birne sind etwa gleich schwer. Oder. Das Buch A kostet genausoviel wie Buch B.) Die Reduktion der Betrachtung einer Sache auf ein oder wenige Markmale nennt man Abstraktion.
- Ähnlichkeit ist nicht notwendigerweise transitiv.
  (Torsten und Sabine sind beide 6 Jahre alt. Sabine und Claudia mögen beide Puppen. Eine Ähnlichkeit zuwischen Torsten und Claudia kann nicht geschlussfolgert werden, da Torsten und Sabine hinsichtlich eines anderen Aspektes (hier Alter) als Sabine und Claudia (Spaß an Puppen) verglichen worden waren.)
- Ähnlichkeit muss nicht symmetrisch sein.
  (Bernd sieht aus wie ein Besen. Doch sehen Besen nicht grundsätzlich aus wie Bernd. Das Problem sind Unschärfen.)

(2.) Die formale Axiomatisierung der inhaltlichen Bedeutung
Jetzt werden die Unschärfen entfernt und alles in klare Strukturen gegossen:
Definition: **Ähnlichkeit**
X ist zu Y ähnlich, gdw. x als Teil von X und y als Teil von Y, in eine Abstraktionsabbildung eingespeist, Bilder hervorbringen, die zueinander isomorph sind. $A(x) \cong A(y)$.

- Aussagenlogisches Prädikat: istÄhnlich(x,y) = { true | false }
- Ordnungsrelation: R(x, y, z) x ist mindestens genauso ähnlich zu y wie zu z.

2.2.3 Die Ähnlichkeit messen

Jede Abbildung $sim : M \times M \longrightarrow [0, 1]$ über eine Menge M verdient nur dann die Bezeichnung „Ähnlichkeitsmaß“ , wenn für alle $x, y, z \in M$ gilt:

- $sim(x, x) = 1$ (Reflexivität)
- $sim(x, y) = sim(y, x)$ (Symmetrie)

- $sim(x, y) = 1 \Leftrightarrow x = y$

2.2.4 Die Distanz messen

Jede Abbildung $d : M \times M \longrightarrow R^+ \cup \{0\}$ über eine Menge M verdient nur dann die Bezeichnung „Distanzmaß“ , wenn für alle $x, y, z \in M$ gilt:

- $d(x, x) = 0$ (Reflexivität)
- $d(x, y) = d(y, x)$ (Symmetrie)
- $d(x, y) = 0 \Leftrightarrow x = y$

2.2.5 Äquivalenz von Distanz und Ähnlichkeit

Wie die beiden Definitionen oben erahnen lasssen, sollte es eine Möglichkeit geben das Distanzmaß in das Ähnlichkeitsmaß zu überführen und umgekehrt. Aber wie? Dazu gibt es 2 Ansätze.

(a) **Die Findung einer Ordnungs-Relation über M (unsere Fälle), die man mit beiden Funktionen realisieren kann.**
Formal steht also folgende Forderung:

$$R_d(x, y, z, s) \Longleftrightarrow R_{sim}(x, y, z, s) \tag{1}$$

R_d kann man wie folgt realisieren:

$$R_d(x, y, z, s) \ g.d.w. \ (d(x, y) \leq d(z, s)) \tag{2}$$

R_{sim} entsprechend:

$$R_{sim}(x, y, z, s) \ g.d.w. \ (sim(x, y) \geq sim(z, s) \tag{3}$$

Distanzmaß und Ähnlichkeitsmaß sind genau dann **kompatibel**, wenn beide Ordnungsrelationen ($R_d(x, y, z, s)$ und $R_{sim}(x, y, z, s)$)) die Menge M auf genau die selbe Weise sortieren. (Siehe (1).)

(b) **Die Findung einer bijektiven, ordnungsinvertierenden Abbildung**

- bijektiv, damit sowohl das Distanzmaß in das Ähnlichkeitsmaß überführt werden kann als auch umgekehrt herum

- ordnungsinvertierend, um die zueinander umgekehrt proportionalen Maße zu versöhnen; d.h. je **kleiner** die Distanz, um so **größer** ist die Ähnlichkeit.

Eine Funktion solcher Gestalt: $f : R^+ \cup \{0\} \longrightarrow [0, 1]$ mit:

- $f(0) = 1$
- $f(d(x, y)) = sim(x, y)$

Beispielsweise erfüllen folgende Funktion unsere Kriterien:

- $f(x) = 1 - \frac{x}{x+1}$
- $f(x) = 1 - \frac{x}{max}$ max entspricht dem maximalen Abstand

2.2.6 Distanzmaß und Metrik

Besitzt ein Distanzmaß $d(x, y)$ auf einer Menge M folgende zusätzliche Eigenschaft:

$$d(x, y) + d(y, z) \geq d(x, z) \quad (Dreiecksgleichung)$$

... so nennt man d eine Metrik und $(M, d(x, y))$ ein metrischer (euklidischen) Raum. Sind die Elemente $m \in M$ je k-dimensional, so besteht die Metrik aus geordneten k-Tupeln reeller Zahlen, welche wir formal so notieren wollen:

$$x = (x_1, ..., x_k) \text{ und } y = (y_1, ..., y_k)$$

Einige Beispiele für Distanzmaße des metrischen Raumes

- Abstand nach Euklid:

$$d(x,y) = \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

- gewogener Abstand nach Euklid:

$$d(x,y) = \sqrt{\sum_{i=1}^{k} w_i(x_i - y_i)^2}$$

- Manhatten Distanz:

$$d(x,y) = \sum_{i=1}^{k} \mid x_i - y_i \mid$$

- Maximum Distanz:

$$d(x,y) = \max_{i=1...k} \mid x_i - y_i \mid$$

- Minkowski Distanz:

$$d(x,y) = \sqrt[r]{\sum_{i=1}^{k} \mid x_i - y_i \mid^r} \qquad r \geq 1$$

2.3 Spezialfall: Metriken mit binären Daten

Nun entsprechen unsere k-Tupel: $x = (x_1, ..., x_k)$ und $y = (y_1, ..., y_k) \in \{0, 1\}$. In diesem Raum entspricht die Manhatten-Distanz der sog. Hamming-Distanz,

$$H(x,y) = k - \sum_{i=1}^{k} x_i y_i - \sum_{i=1}^{k}(1 - x_i)(1 - y_i)$$

welche die Anzahl der voneinander verschiedenen Bits (x_i zu y_i) aufsummiert.

2.3.1 Differenziertere Ergebnisse durch Kontingenztafel

Betrachtet man zwei k-Tupel, z.B. $x = (001011)$ und $y = (011001)$ hinsichtlich ihrer Ähnlichkeit, lassen sich insbesondere vier statistische Aussagen ableiten:

1. Die Anzahl **a** der Stellen wo x_i und y_i beide den Wert 1 haben.

$$a = \sum_{i=1}^{k} x_i y_i$$

   In unserem Beispiel: a=2.

2. Die Anzahl **b** der Stellen wo $x_i = 0$ und $y_i = 1$ ist.

$$b = \sum_{i=1}^{k} (1 - x_i) y_i$$

   In unserem Beispiel: b=1.

3. Die Anzahl **c** der Stellen wo $x_i = 1$ und $y_i = 0$ ist.

$$c = \sum_{i=1}^{k} x_i (1 - y_i)$$

   In unserem Beispiel: c=1.

4. Die Anzahl **d** der Stellen wo x_i und y_i beide den Wert 0 haben.

$$d = \sum_{i=1}^{k} (1 - x_i)(1 - y_i)$$

   In unserem Beispiel: d=2.

Es gilt: $k = a + b + c + d$. Diesen Zusammenhang kann man in einer Tabelle darstellen → dann „Die Kontingenztafel“ genannt.

2.3.2 Der Simple-Matching-Coefficient (SMC)

Wie baut man sich einen SMC? Kurzanleitung – so geht's:

(a) Mit den Ergebnissen der Kontingenztafel kann man den Hammingabstand auch mi Hilfe von a, b, c, und d darstellen. Da b und c die Anzahl der verschiedenen Bits darstellt, ergibt sich:

$$H(x,y) = b + c = k - (a + d)$$

(b) Nun transformieren wir die Hamming-Funktion in ein kompatibles Ähnlichkeitsmaß und dies, unter Zuhilfenahme der Transformationsfunktion $f(x) = 1 - \frac{x}{max}$. Wobei max der Anzahl der Stellen entspricht, die sich wiederum aus der Dimension k unseres Tupels ergibt.

$$f(H(x,y)) = 1 - \frac{H(x,y)}{k} \quad (4)$$

$$sim_H(x,y) = 1 - \frac{b+c}{k} \quad (5)$$

$$sim_H(x,y) = \frac{a+d}{k} \quad (6)$$

$$sim_H(x,y) = 1 - \frac{b+c}{k} \quad (7)$$

(c) Fertig! Doch gibt es einige Besonderheiten(!!!):

$$sim_H((x_1,...,x_k),(y_1,...,y_k)) \neq sim_H((1-x_1,...,1-x_k),(1-y_1,...,1-y_k))$$

sim hängt in unterschiedlicher Weise von den einzelnen Attributen (a, b, c, d) ab.

2.3.3 Optimistische und pessimistische Strategien (SMC)

Interessant ist nun die Frage: Wie kann man erreichen, dass bei der Ähnlichkeitsmessung übereinstimmende Merkmale stärker/schwächer ins Gewicht fallen als nicht übereinstimmende solche?
Diese Manipulation kann durch ein Gewichts-Parameter bewirkt werden: α mit $0 \leq \alpha \leq 1$. Damit ergibt sich ein solcher modifizierter SMC:

$$sim_\alpha(x,y) = \frac{\alpha(a+d)}{\alpha(a+d) + (1-\alpha)(c+b)}$$

Für $\alpha = \frac{1}{2}$ erhält man den normalen SMC. Zur Erinnerung: $k = a+b+c+d$.

Nun lassen sich daraus ein paar interessante Aussagen ableiten. Je nach Wahl von α können folgende Bewertungskriterien unterschieden werden:

$\alpha > \frac{1}{2}$: Übereinstimmende Attribute werden höher bewertet als die zueinander verschiedenen (optimistische Interpretation)
$\alpha = \frac{1}{2}$: Übereinstimmende und verschiedene Attribute werden gleich bewertet.
$\alpha < \frac{1}{2}$: Übereinstimmende Attribute werden niedriger bewertet als die zueinander verschiedenen (pessimistische Interpretation)

2.3.4 Weitere Modifikationen an SMC's

In den letzten 70 Jahren haben verschiedene Wissenschaftler weitere, oft verwandte Modifikationen vorgenommen:

Zunächst die symmetrischen:

$$sim(x,y) = \frac{a+d}{a+d+2(b+c)} \qquad \text{Rogers/Tanimoto 1960}$$

$$sim(x,y) = \frac{(a+d)-(b+c)}{a+b+c+d} \qquad \text{Hamman 1961}$$

$$sim(x,y) = \frac{a+d}{2(a+d)+b+c} \qquad \text{Sokal/Sneath1963}$$

Die asymmetrischen:

$$sim(x,y) = \frac{2a}{2a+b+c} \qquad \text{Sorensen 1948}$$

$$sim(x,y) = \frac{1}{2}\frac{a}{a+b} + \frac{a}{a+c} \qquad \text{Kulczynski 1927}$$

$$sim(x,y) = \frac{a}{a+2(b+c)} \qquad \text{Sokal/Sneath1963}$$

$$sim(x,y) = \frac{a+d}{\sqrt{(a+b)(a+c)}} \qquad \text{Ochial 1957}$$

2.4 Erweiterung von Distanz- und Ähnlichkeitsmaßen

Jetzt wollen wir uns kurz ein paar "Verfeinerungen"dieser Maße ansehen, um auch sehr ausdifferenzierte Strukturen zu bewältigen.

2.4.1 Berücksichtigung von Attribut-Alternativen

Manchmal will man auch die Anzahl der **möglichen Attributwerte** in das Ähnlichkeitsmaß einfliessen lassen. Übereinstimmung bei einem Attribut mit vielen Ausprägungen (z.B. Wert zwischen 1 und 1000) könnte höher zu bewerten sein, als die Übereinstimmung bei einem Attribut, das nur wahr oder falsch sein kann.
Ein solches Maß könnte beispielsweise folgendermaßen aussehen:

$$sim(x, y) = \frac{1}{m} \sum_{i=1}^{n} m_i \delta(x_i, y_i)$$

mit m_i als Anzahl der möglichen Ausprägungen des i-ten Attributs und $m = \sum_{i=1}^{n} m_i$. δ ist bestimmt als

$$\delta = \begin{cases} 1 \text{ falls } x_i = y_i \\ 0 \text{ sonst} \end{cases}$$

2.4.2 Attribute mit Gewichten

Man kann jedem Attribut x_i ein Gewicht w_i zuordnen, mit $\sum w_i = 1$. Das Problem hierbei ist eine sinnvolle Abschätzung der Wichtung; hierbei kann u.U. zur Verschlechterung der Güte des gesamten System kommen.

2.5 Unbekannte Attribute

Manchmal sind für die Berechnung der Ähnlichkeitsmaße nicht alle Attribut-Werte bekannt (z.B. es wurde ein Fragebogen nicht vollständig ausgefüllt). Was tun? Hier einige, teilweise einfältige Ansätze:

- Jedes vohandene Attribut erhält einen **"Jubelpunkt"**, welche mit aufsummiert werden.
- Man **normiert** das Maß bezüglich der Anzahl vorhandener Attribute.

- Nur Attribute, welche in **beiden Tupeln vorhanden** sind werden berücksichtigt.
- Fehlende Werte werden **interpoliert**. Optimistisch: Man nimmt an, dass die Werte gleich sind. Pessimistisch: Verschiedenheit der Werte wird unterstellt.

2.5.1 Wir sind das Universum — Closed-World Assumption (CWA)

Nehmen wir an, dass das, was wir über die Welt wissen, vollständig ist. Unsere Wissensbasis enthält alles was wahr und ableitbar ist. Kommt nun ein Faktum - in Form eines Literales L - an uns vorbei, und ist dies weder Teil unserer Wissensbasis, noch lässt es sich aus ihr heraus herleiten, dann beschließen wir, dass es L nicht gibt und merken uns das in unserer Wissensnbasis. Folmal:

$$\text{Wenn } \neg(Q \vdash L), \text{ dann betrachte Q so, als gelte } Q \cup \{\neg L\}.$$

Lies: Wenn man L nich von Q ableiten kann, dann füge $\neg L$ zur Menge Q (die Wissensbasis) hinzu[1].

- CWA ist nicht monoton und kann zu Inkonsistenzen führen
- ist jedoch für Horn-Klauseln konsistent

2.5.2 Default-Logik Erweiterung

Die Default-Logik-Erweiterung stellt ein Mittel bereit, um „schwache“ Inferenzregeln. Man kann mit ihr Aussagen der Art „Typischerweise gilt...“ treffen. Wie schaut solch ein Default nun aus?

$$\delta = \frac{A(X) : B_1(X), ..., B_n(X)}{C(X)}$$

mit

- $A(X)$: Vorbedingung (X muss aus der Datenbasis ableitbar sein!)
- $B_i(X)$ Rechtfertigungen des Defaults

[1] Diese Regel läßt sich mit dem Erweiterungs-Lemma der Prädikatenlogik rechtfertigen: Läßt sich einer konsistenten Theorie Q eine Formel L nicht ableiten, dann ist auch die Erweiterung $Q \cup \{\neg L\}$ von Q um $\neg L$ eine konsistente Theorie.

- $C(X)$ Konsequenz

Ein Beispiel:
„Vögel fliegen typischerweise“

$$\delta_1 = \frac{Vogel(x) : hatFedern(x) \wedge kannFliegen(x)}{kannFliegen(x)}$$

Erklärung: Wenn x also ein Vogel ist und die Annahme $kannFliegen(x)$ möglich ist, **ohne zu Inkosistenzen zu führen**, dann nimm $kannFliegen(x)$ an. Ausnahmen werden durch Regeln der Art dargestellt:

$$\forall x : Pinguin(x) \longrightarrow \neg kannFliegen(x)$$

2.5.3 Attribute und Häufigkeiten

Angenommen unsere Fall-Datenbak ist gefüllt mit so mancherlei Fällen, die wiederum etliche Attribute besitzten. Da könnte es doch sein, dass bestimmte Attribut-Werte sehr häufig vorkommen, also viele Fälle z.B. das Attribut „*binGlücklich*“ mit dem Wert „ja ($true$)“ besitzen. Sollte nun ein Fall „*unglücklich*“ sein, so sollte ihm besondere Aufmerksamkeit zukommen, d.h. die Wichtung dieses Falles muss besonders schwer ausfallen. Aber auch der umgekehrte Fall ist vorstellbar. Es ist also vor dem Anwenden des Ähnlichkeitsmaßes eine kleine statistische Erhebung über die Fall-Basis notwendig, um diese Besonderheiten mit in die Ähnlichkeisbestimmung einfließen zu lassen.
Stellen wir uns hierfür ein Werkzeug zur Verfügung. Sei H_{i0} die Häufigkeit des Nicht-Auftretens des i-ten Attributs, H_{i1} entsprechend die Häufigkeit des Auftretens. Sei weiterhin n die Anzahl unserer Fälle in der Fallbasis.
Es gilt:

$$H_{i0} = \sum_{j=1}^{n} (1 - x_j(i))$$

$$H_{i1} = \sum_{i=1}^{n} (x_j(i))$$

Da ja nun selteneres Auftreten von Übereinstimmungen höher bewertet werden soll, werde diese Übereinstimmungen mit der Häufigkeit der Nicht-Übereinstimmungen gewichtet und umgekehrt. Es ergeben sich also folgende gewichtete Werte von a und d:

$$\hat{a} = \frac{1}{n} \sum_{i=1}^{k} x_i \cdot y_i \cdot H_{i0}$$

$$\hat{d} = \frac{1}{n} \sum_{i=1}^{k} (1 - x_i) \cdot (1 - y_i) \cdot H_{i1}$$

2.6 Aggreagation von Ähnlichkeitsfunktionen

Das ist eine völlig abgedrehte Idee, basierend auf der Tatsache, dass verschiedene Ähnlichkeitsfunktionen unterschiedliche Aspekte mehr oder weniger berücksichtigen.

2.6.1 Geschachtelte Funktionen

„Warum schachteln wir diese Funktionen nicht mal ineinander? “ dachte sich irgendwer - und schon war es geschehen. Und das geht so:

$$sim_b(\vec{x}, \vec{y}, \vec{sim}) = sim_a(sim_1(x_1, y_1), sim_2(x_2, y_2), ..., sim_k(x_k, y_k))$$

Wobei sim_1 bis sim_k jetzt nur noch einzelne Attribut-Werte behandeln - keine Vektoren von Werten. Somit können jedem Attribut eigene Ähnlichkeitsfunktionen zugeordnet werden. Dies steigert in der Tat die Qualität der Ähnlichkeitsbestimmung.

2.6.2 Aufsummierte Ähnlichkeitsfunktionen

Interessant ist auch die Möglichkeit die Ähnlichkeitsfunktion im geometrischen Mittel aufzusummieren:

$$sim_b(\vec{x}, \vec{y}, \vec{sim}) = \frac{1}{k} \sum_{i=1}^{k} sim_i(x_i, y_i)$$

Bemerkung: Die Suche nach dem ähnlichsten Fall ist eine Optimierungsaufgabe: $sim(x, y) \longrightarrow \max$

2.7 Taxonomien

... sind Ähnlichkeitsmaße als Distanz zweier Objekte (besser Knoten) in einem Graphen, der durch einen zyklenfreien Baum dargestellt wird. Wichtig sind hierbei Knoten, als Träger der Attribute und Kanten mit einer Kantenlänge als Abstandsmaß.

3 Beispiel: Ein fallbasiertes Inferenzsystem

Vorstellbar ist ein System zur Unterstützung medizinischer Diagnostik. Solch ein System muss besonderen Anforderungen genügen:

- Bestimmte Attribute müssen für verschiedene Diagnosen unterschiedlich stark brücksichtigt werden können.
- Es muss zwischen pathologischen[2] und normalen Attributen unterschieden werden.
- Unterschiedliche Attribute sollten nicht immer zwangsläufig zu einer Abwertung des Ähnlichkeitsmaßes führen.

3.1 Relevanz von Attributen

Wie aber unterscheiden wir nun zwischen den für den speziellen Fall relevanten und den weniger relevanten Attributen? Zudem kann es sein, dass beim Zugriff auf mehrere Datenbasen (eine in München, die andere in Dresden) redundante oder gar falsche Informationen zusammengetragen und ausgewertet werden. Diese (insbesondere die redundanten) dürfen die Güte der Ähnlichkeit nicht negativ beeinflussen.
Hier gibt es einige mehr oder weniger triviale Ansätze:

(a) implizit konsistente Daten - d.h. nur sinnvolle Fallbeispiele eingeben :-)

(b) Lernen mit symbolischen Ansätzen

(c) Attribute mit Gewichten versehen

[2] pathologisch = krankhaft, abnormal

3.1.1 Globale Wichtungen

Alle Attribute eines Falles $x = (x_1, ..., x_k)$ erhalten eine Wichtung $w = (w_1, ..., w_k)$. Global meint in diesem Fall, dass alle Attribute x_i unabhänngig von der betrachteten Diagnose d_j. D.h. für jedes Attribut existiert genau ein Gewichts-Faktor. Probleme bei globaler Wichtung: Ergebnisse sind total unbefriedigend!

3.1.2 Lokale Wichtung – die Relevanzmatrix

Nun erhalten alle Attribute x_i eine Wichtung w_{ij}, welche nun von einer speziellen Diagnose (Teil eines Falles) d_j abhängt[3].

Wie speichern wir die vielen Gewichte? Genau - in einer Matrix! Wir definieren uns eine $m \times k$ Relevanzmatrix, wobei k die Anzahl der definierten Attribute und m die Zahl der unterschiedlichen Diagnosen d_j vertritt.

	d_1	d_2	...	d_m
x_1	w_{11}	w_{12}	...	w_{1m}
x_2	w_{21}	w_{22}	...	w_{2m}
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
x_k	w_{k1}	w_{k2}	...	w_{km}

Ein Spaltenvektor $\vec{d_j} = (w_{1j}, ..., w_{k,j})$ beschreibt damit die Relevanz der Attribute unter der Diagnose d_j. Wichtig ist zudem, dass der Vektor normiert ist, so dass gilt:

$$\sum_{i=1}^{k} w_{ij} = 1$$

3.1.3 Bestimmung der Relevanzmatrix

Wie aber kommen wir zu den konkreten Werten in unserer Matrix? Dazu nun einige Ansätze:

- automatische Bestimmung $\longrightarrow$ mit Hilfe von bekannten Fällen - Häufigkeitsanalyse

[3]Wollen wir nicht alle gern mal ein Stündchen abhängen?

- fragen wir ein paar Experten und tragen die Wichtigkeit per Hand ein
- Benutzen wir die Fall-Basis als Trainigsmenge und trainieren/überprüfen die Daten

Initialisierung der Matrix

Hier ist der erste Ansatz sehr nützlich. Besorgen wir uns von lieben Kollegen eine Trainigsmenge von Fällen. Sei h_{ij} nun die Häufigkeit des Auftretens der Attributausprägung x_i unter der Diagnose d_j. *Beispielsweise sei die Diagnose „Schnupfen“ . Dann gibt es verschiedene Fälle von Schnupfen. Mal mit, mal ohne Gliederschmerzen, mal mit hohem Fieber u.s.w..* Nun nutzt man die Häufigkeit, um die Matrix wie folgt zu initialisieren:

$$w_{ij} = \frac{h_{ij}}{\sum_{i=1}^{k} h_{ij}}$$

3.1.4 Veränderung der Relevanzmatrix durch Lernen

Was wir brauchen sind einige hundert Testdaten (Trainingsmenge), die je aus einem Tupel $(\vec{x}, d)$ bestehen, also aus einem Attribut-Vektor $\vec{x}$ und einer in Bezug auf $\vec{x}$ korrekten Diagnose d. Nun nehmen wir uns ein solches Tupel $(\vec{x}_{akt}, d_{akt})$ heraus, legen es dem System vor, welches den ähnlichsten Fall $(\vec{x}, d)_{ähn}$ heraussucht. Es gibt folgende Möglichkeiten des Vorgehens:

(1.) $d = d_{akt}$, dann tue nichts.

(2.) $d \neq d_{akt}$, jetzt ergeben sich 2 Möglichkeiten

  (a) $\vec{x} \subseteq \vec{x}_{ähn}$; der Fall $(\vec{x}, d)_{ähn}$ wird durch das Trainingstupel $(\vec{x}_{akt}, d_{akt})$ ersetzt.

  (b) $\vec{x} \nsubseteq \vec{x}_{ähn}$, d.h. der Fall $(\vec{x}, d)_{ähn}$ bleibt in der Fallbasis (FB), der neue Fall wird aufgenommen, die Gewichte ändern sich unter folgenden Maßgaben:

    * Normierung beachten: $\sum_{i=1}^{k} w_{ij} = 1$
    * redundante Attribut-Werte bekommen keine Gewichte
    * Die Gewichte gleicher Attribute mit unterschiedlichen Werten werden erhöht.
    * Die Gewichte gleicher Attribute mit gleichen Werten werden erniedrigt.

3.1.5 Bewertung redundanter Attribute auf Basis der Abnormalität

Bisher wurden redundante Attribute bei der Bewertung durch die Gewichte in unserer Matrix nicht berücksichtigt. Jedoch kann eine Bewertung dieser Attribute sinnvoll sein, insbesondere wenn es sich um pathologische handelt. Hier sollte eine Änderung/Abwertung des Ähnlichkeitswertes erfolgen. Man kann die Abnormalität grob in zwei Klassen unterteilen:

- Kontextunabhängige Abnormalität: eine Attributausprägung kann in allen Situationen als pathologisch bezeichnet werden.
- Kontextabhängige Abnormalität: die Attributausprägung kann nur in einem bestimmten Zusammenhang als pathologisch bezeichnet werden.

Diese Informationen müssen in geeigneter Weise in die Wissensbasis einfließen. Z.B. über die Schaffung einer Liste mit Regeln, die über die Wissensbasis Anwendung finden. Diese Regeln liegen dann in z.B. folgender allgemeiner Form vor:

- $\vec{x} \longrightarrow (x_i = v_i)$
  Lies: Wenn das konkrete Attribut x_i des Attributvektors $\vec{x}$ den Wert v_i hat, dann wird diese Ausprägung als pathologisch bezeichnet.
- Kontextunabhängige pathologische Attributausprägungen werden durch eine leere Fallmenge charakterisiert:
  $\{\} \longrightarrow (x_i = v_i)$

3.1.6 Ähnlichkeit auf Attributen

Bisher haben wir immer die Ähnlichkeit von Fällen bezogen auf ihre Aspekte betrachtet, also die Anzahl übereinstimmender bzw. abweichender Attributwerte. Manchmal kann es jedoch auch sinnvoll sein, einzelne Attributwerte miteinander zu vergleichen (z.B. Schmerz stark - Tank voll, o.ä.). Hierfür benötigen wir weitere Ähnlichkeitsmaße, die lokal für jedes Symptom S_i bzw. jeden Wertebereich V_i definiert sind. Die uns bekannten Symptomtypen sind

1. binär
   (a) symmetrisch: $istBlau(x) = \{true|false\}$ [true und false sind gleichwertig]
   $$w(v_i, v_j) = \begin{cases} 1 & \text{falls } v_i = v_j \\ 0 & \text{sonst} \end{cases}$$

(b) asymmetrisch: Spannung-12V? = (ja | nein) [ja wiegt stärker]

$$w(v_i, v_j) = \begin{cases} 1 & \text{falls } v_i = v_j = 1 \\ c \in [0,1] & \text{falls } v_i = v_j = 0 \\ 0 & \text{sonst} \end{cases}$$

2. mehrstufig

   (a) geordnet: (komperativ) (groß, größer, riesig)
   Falls eine totale Ordnung existiert, also

   $$[v_1, v_2, \ldots, v_n] \text{ mit } v_i \prec v_j \Leftrightarrow i < j$$

   so kann man mit $ord(v_i) = i$ ein Ähnlichkeitsmaß wie folgt definieren

   $$w(v_i, v_j) = 1 - \frac{|ord(v_i) - ord(v_j)|}{n}$$

   (b) ungeordnet: (klassifikatorisch) (blau, rot, gelb)
   Da keine Ordnungsrelation auf den Werten existiert, muss für jedes mögliche Wertpaar (v_i, v_j) eine Ähnlichkeit c_{ij} definiert werden. Man erhält dann

   $$w(v_i, v_j) = \begin{cases} 1 & \text{falls } v_i = v_j \\ c_{ij} & \text{sonst} \end{cases}$$

3.2 Instanzbasiertes Lernen

3.2.1 Instance-based lerning Algorithm

Dieser Algorithmus stellt eine spezielle Form des fallbasierten Lernens dar, bei welchem nur die Fallbasis (FB) angepasst wird, das Ähnlichleitsmaß hingegen bleibt konstant. Zudem ist eine geometrische Interpretation der Fallbasis notwendig. So ist jeder Fall ein Punkt in einem k-dimensionalen Raum und der Abstand d sei der gewöhnliche euklidische Abstand:

$$d(x, y) = \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2}$$

Die Fallbasis weist nun zweierlei Strukturen auf:

- Einen Abstand der Punkte (Attributvektor der Fälle) im Raum (FB) nach Euklid
- Klassen von Diagnosen d

Für das Training wird eine Trainingsmenge T von voher ausgewählten Testfällen t verwendet, wobei $t \in T$ immer aus Problem p_t und Lösung l_t besteht.

3.2.2 Der Algorithmus (IB1)

Solange nicht alle $t \in T$ betrachtet wurden, nehme t in Fallbasis auf. Recht einfach, oder? ;-)
Problem hierbei ist natürlich, dass die FB extrem groß wird und auch redundante Fälle aufgenommen werden.

3.2.3 Der Algorithmus (IB2)

Solange nicht alle $t \in T$ betrachtet wurden, tue folgendes:

1. Wähle nächsten Nachbarn t_n zu t aus der FB
2. Falls $l_{t_n} \neq l_t$ nimm t in FB auf

IB2 speichert wesentlich weniger Fälle, als IB1 und ist dennoch nur unwesentlich schlechter in der späteren Klassifikation. Welche Fälle aufgenommen werden hängt von der Präsentationsreihenfolge der Testfälle ab! Leider führen verrauschte Daten oft zu Klassifikationsfehlern.

3.2.4 Der Algorithmus (IB3)

Hierfür brauchen wir noch ein statistisches Maß für die Güte der Klassifizierung. Es ist bestimmt durch

$$KG(F) = \frac{\text{Anzahl der korrekt klassifizierten Probleme}}{\text{Anzahl aller klassifizierten Probleme}}$$

Arbeite genauso wie bei IB2 und entferne hinterher den nächsten Nachbarn, so er statistisch gesehen zu Fehlklassifikationen bei verrauschten Daten führt.

4 Semantische Netze

4.1 KL-ONE beschreibt die Welt

Die Ideen zu KL-ONE stammen von Brachmnann (1977) und basieren auf einer objektorientierten, deklarativen Wissensdarstellung mit Hilfe von Mengen, Begriffen und Aussagen, die durch ein logisches Netz semantisch untereinander verknüpft werden.

4.1.1 Begriffe und Definitionen

Konzept

- **primitive** Konzept: z.B. Mensch
- **definierte** Konzept: z.B. Junggeselle
- **individuelle** Konzept: z.B. Papst (dieses Konzept ist einmalig)

Rolle

- Relation zwischen Konzepten
- mit möglicher Einschränkung des Wertebereiches (Range -Konzept)

Subsumtion vgl. Vererbung

- A subsumiert B, wenn jede Instanz von B auch Instanz von A ist.

Terminologie

- Menge von Gleichungen und Ungleichungen zwischen entweder Rollen oder zwischen Konzepten

Assertionen

- Füllt definierte Konzepte und Rollen mit konkreten Individuen.
- Dafür gibt es sog. assertionale Axiome, auch ABox genannt.
- **Beispiele:**
  Elisabeth: Frau
  Elisabeth hat-kind Beder

TBox: endliche zyklenfreie Teilmenge von Konzepten und Rollen

- Jeder Konzept- oder Rollenterm darf höchstens einmal auftreten.

- Beispieel:
  Frau $\subseteq$ Mensch (ist Untermenge - oder: erbt von)
  Vater = Mann $\cap\exists$ hat-kind.Mensch

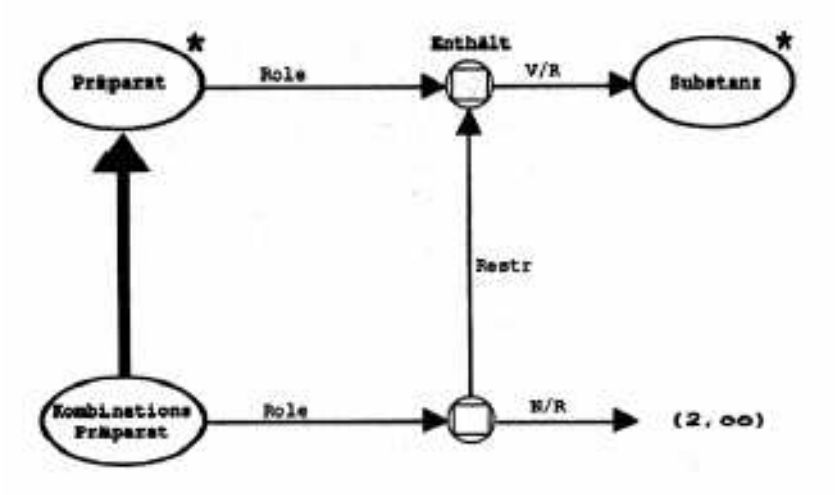

Abbildung 3: Präparat und Substanz sind primitive Konzepte, das Kombinationspräparat ist ein definiertes Konzept. Ein Präparat enthält eine beliebige Anzahl von Substanzen. Ein Kombinationspräparat ist ein Präparat, das mind. 2 Substanzen enthält.

4.1.2 Entscheidende Schwächen

- Probleme bei komplexen Objekten
- closed word assumption
- statische Wissensrepresentation
- keine Abbildungen zeitabhängiger Attribute

5 Indexing und Retrieval

Die Situation: Wir haben eine hinreichend große Fallbasis und wollen ein wenig mit ihr arbeiten. Dazu nehmen wir uns einen Problemfall her und suchen nach ähnlichen Fällen in unserer Fallbasis. Ist die Bsais groß, ergibt sich ein massives Suchproblem. Ist keine exakte Übereinstimmung (exact match) auzumachen, muss die ähnlichste Lösunge (best match) gefunden

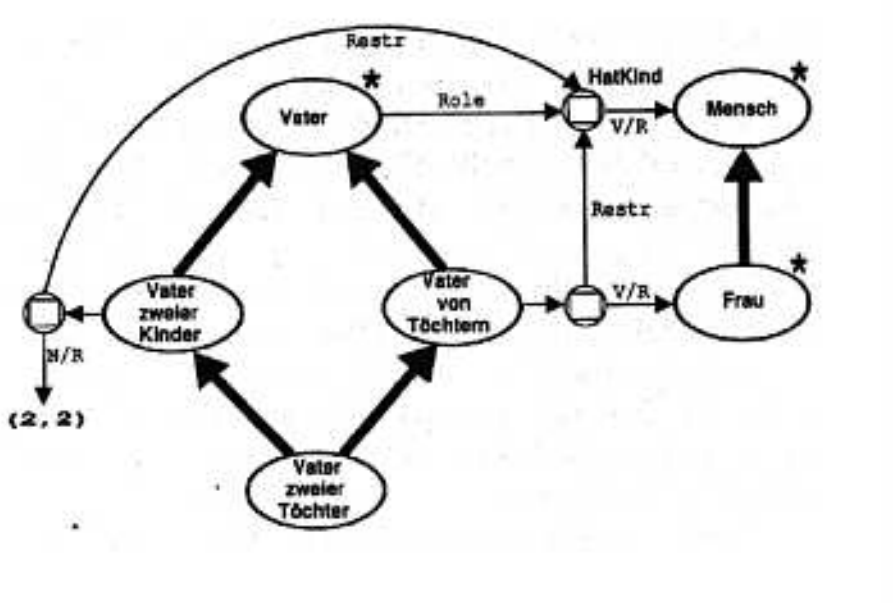

Abbildung 4: Hier sieht man die Restriktion von Rollen durch Kardinalitätsangaben: hat genau zwei Kinder. Weiter haben wir eine Restriktion durch Wertebeschränkung: hat Kind, hat Tochter

werden, dann aber ist auch eine Anpassung der gefundenen Lösung an die gegebene Situation durchzuführen.

5.1 Indexierung von Fällen

Um das erste Problem des Suchens zu beschleunigen, hilft die Einführung eines Indexes. Dieser nummeriert nicht etwa die Fallbasis durch - das wäre wenig hilfreich, sondern benutzt bestimmte relevante Merkmale eines Falles.

> *Vorstellbar wäre z.B. folgende Indizierung für einen schweren Schnupfen:* [*Nase-läuft*].[*Augen-geschwollen*].[*Fieber*]
> *Es werden also die Attribute in einer bestimmten Reihenfolge benutzt um einen Index zu erzeugen.*

Nun ist der Zugriff erheblich effizienter[4] und der Index arbeitet wie ein Filter. Leider ergeben sich daraus auch ein paar **Probleme:**

- Es ist schwer die richtigen Index-Attribute auszumachen und dann auch noch in einer Reihenfolge festzulegen.

[4] Effizienz $= \frac{Erreichbarkeit eines Zieles}{Zeit}$.

- Der Index determiniert den Lösungsraum. Es kann passieren das richtige Lösungen ausgefiltert werden.
- Attribute, welche nicht in den Index einfließen, tragen überhaupt nicht mehr zur Lösung bei.

Wir brauchen also eine Index-Struktur, welche sich nicht allein auf ausgewählte Attribute gründet, sondern grobe Lösungsziele berücksichtigt (Unschärfe) und Fehler bei vorherigen Versuchen einbezieht. Daran wird heftigst geforscht. Folgende Ansätze können genannt werden:

- starre, definierte Indizierung *(siehe Beispiel)*
- induktive Indizierung (braucht Induktionsregeln)
- Erklärungsbasierte Induzierung ?????
- Induzierung mit Wiedererkennung von Fällen über Teildomänen

5.2 Techniken

5.2.1 Direct Reminding - without indexing

- Berechnung eines Ähnlichkeitsmaßes
- Partitionierung der Fallbasis **zur Laufzeit** in Klassen:
  - K_0: Bewiesen
  - K_1: Ausreichend
  - K_2: Wahrscheinlich
  - K_3: Möglich
  - K_4: Unwahrscheinlich
- Berechnung eines Ähnlichkeitsmaßes nur für die höchste(n) Klasse(n).

5.2.2 Parallel Retrieval

Hier werden aus dem Eigenschaftsvectoren wichtige Aspekte herausgenommen und daraus Sub-Räume gebildet, welche parrallel durchsucht werden.

- **mit gleichzeitigem Vergleich** des aktuellen Falles mit allen Fällen der Fallbasis und Bewertung.

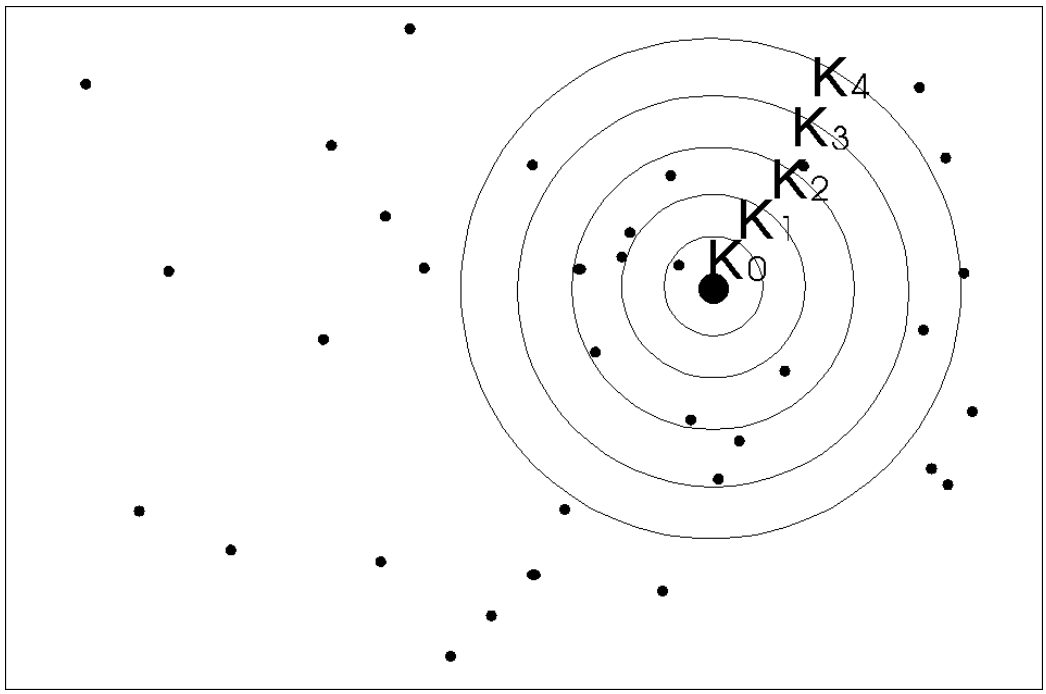

Abbildung 5: Direct Reminding: um den gegebenen Fall werden Klassen-Ringe gezogen und somit der zu durchsuchende Raum eingeschränkt

- **Partielle Übereinstimmungen** werden mit einer speziellen Evaluierungsfunktion auf alle Einträge der FB zugleich angewandt.

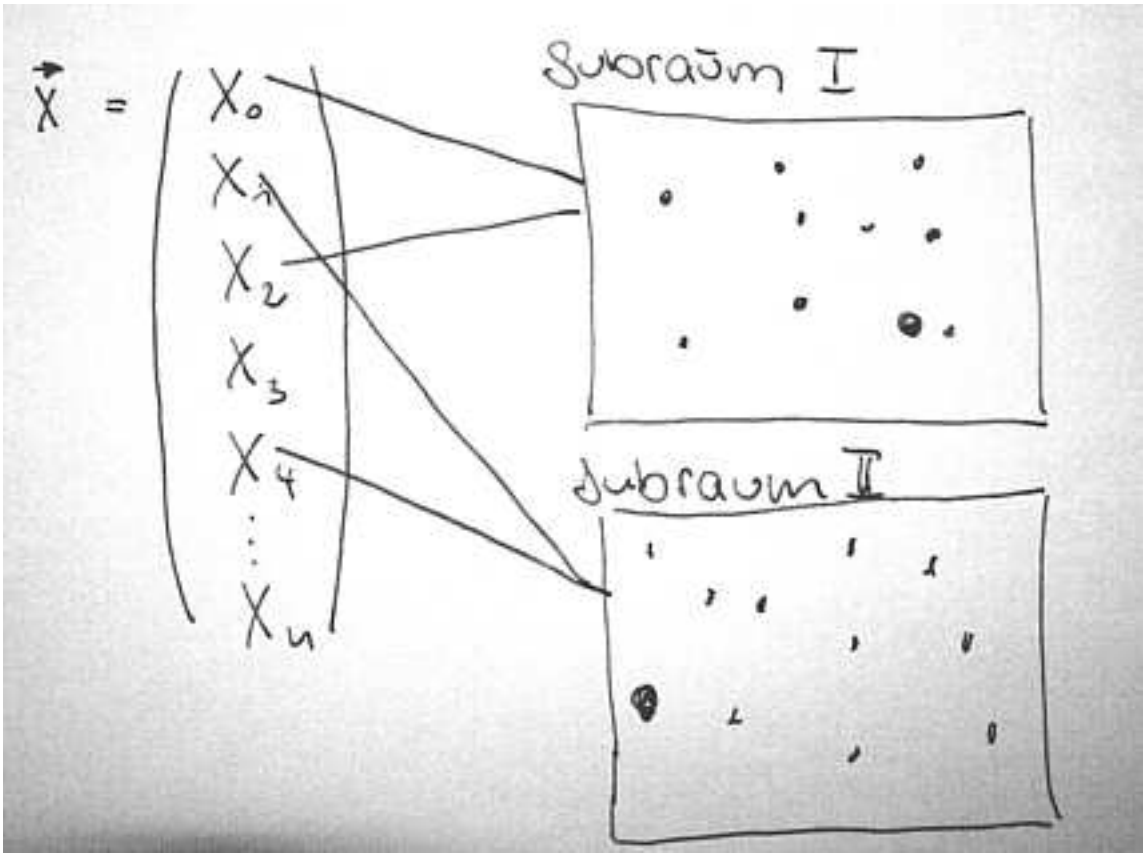

Abbildung 6: Parralel Retrieval: Bildung von Subräumen und parallele (zeitgleiche) Suche

5.2.3 Clustering (k-Means)

Ausgangssitutaion

- Objekte besitzen Distanzfunktion
- für jedes Cluster kann ein Clusterzentrum bestimmt werden („Mittelwert")
- Anzahl k der Cluster wird vorgegeben

Basis-Algorithmus

1 (Initialisierung): k Clusterzentren werden (zufällig) gewählt

2 (Zuordnung): Jedes Objekt wird dem nächstgelegenen Clusterzentrum zugeordnet

3 (Clusterzentren): Für jedes Cluster wird Clusterzentrum neu berechnet

4 (Wiederholung): Abbruch, wenn sich Zuordnung nicht mehr ändert, sonst zu Schritt 2

Probleme

- Konvergenz zu lokalem Minimum, d.h. Clustering muss nicht optimal sein
- Work-around: Algorithmus mehrfach starten

5.2.4 Voronoi-Diagramme

Wie einige Kapitel zuvor schon kurz skizziert, kann man sich die Fallbasis auch als Punkt-Wolke im k-dimensionalen Raum vorstellen, wobei die Attributvektoren der einzelnen Fälle die Punkte beschreiben. Soll nun ein unbekannter (neuer) Fall klassifiziert werden, besteht die Aufgabe im Wesentlichen darin, ihn zu dem bekannten Fall der Basis zuzuschreiben, der ihm am nächsten ist; also den Punkt der vorhandenen Punktwolke zu bestimmen, der dem neuen Fall mit $d(x, y) = CD^2 \longrightarrow R^+$ am nächsten ist. Das Vorinoi-Diagramm ist hierbei ein Hilfsmittel für eine korrekte Klassifizierung. Es setzt sich aus einer Vielzahl von Voronoi-Polygonen zusammen.

Voronoi Polygone Die Konstruktion erfolgt, indem auf allen Verbindungsgraden zwischen den Messpunkten auf der Hälfte eine Senkrechte konstruiert wird und mit den anderen Senkrechten verbunden wird.

Spezielle Eigenschaften Das Voronoi-Diagramm zerlegt die Ebene in disjunkte und konvexe Gebiete.Weitere wesentliche Merkmale sind:

- die Zahl der Kanten verhält sich linear zu der Anzahl der Knoten
- eine Kante trennt die Gebiete zweier benachbarter Punkte zum halben Abstand (Mittelsenkrechte)
- ein Voronoi-Dagramm mit n Punkten **der Ebene** kann in $O(n \log n)$ Schritten konstruiert werden und braucht $O(n)$ Speicher

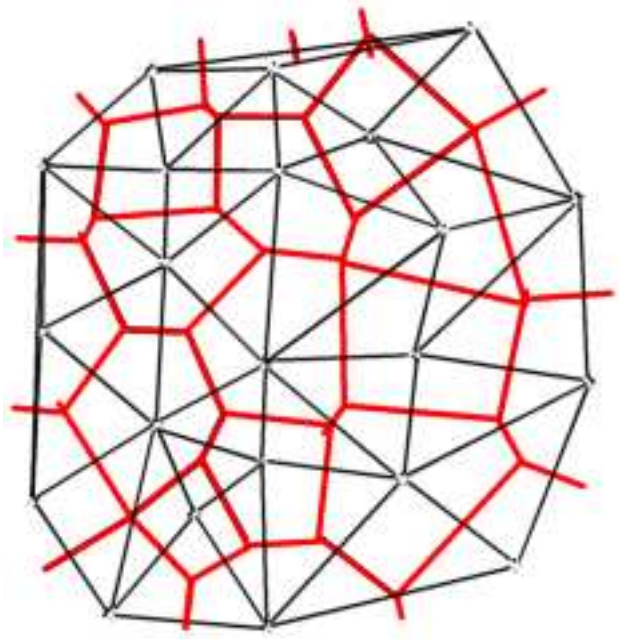

Abbildung 7: Ein aus Polygonen zusammmengesetztes Voronoi-Diagramm

- hat ein solcher Graph n Knoten, so hat er höchstens $3 \cdot n - 6$ Kanten

5.2.5 Multidimensionale binäre Bäume (k-d Trees)

Die Idee beseteht in einer anderen, besonderen Art des Speicherns der Falldaten. Sie werden nunmehr in einer **netzartig**en Struktur als Knoten abgelegt, wobei die **Ähnlichkeit** der Fälle sich **durch die Nähe** der Knoten zueinander ausdrückt. Das ist sehr, sehr spannend! Eine Möglichkeit ist die Abbildung der Daten in einem multidimensionenlen Baum. Beginnen wir zunächst mit einem Beispiel im 2-dimensionalen Raum. Und das geht so:

Aufbau eines k-d Baumes In Prosa:
(A) Betrachte den Raum. Gibt es in ihm genau einen Punkt, dann erkläre diesen zum Blatt und hefte ihn an den Baum.
Du bist am Ziel.
Gibt es hingegen mehrere Punkte, dann schaue zunächst nach, auf welcher Ebene des Baumes wir gerade sind (mit 0 beginnend).

(B1) Ist die Ebenenzahl (die Tiefe also) eine gerade Zahl, dann tue folgendes: teile den Raum in zwei Unter-Räume, indem Du den Mittelwert der x-Koordinaten aller im jetzigen Raum befindlichen Punkte bildest und an dieser Koordinate eine vertikale Linie ziehst. Parallel dazu fügst Du (sozusagen zeitgleich) einen Knoten zum Baum hinzu. Für jeden Unterraum führe seperat den Algorithmus aus: gehe mit dem Unterraum zu **(A)**

(B2) Ist die Ebenenzahl (die Tiefe also) eine ungerade Zahl, dann tue dies: teile den Raum in zwei Unter-Räume, indem Du den Mittelwert der y-Koordinaten aller im jetzigen Raum befindlichen Punkte bildest und an dieser Koordinate eine horizontale Linie ziehst. Parallel dazu fügst Du (sozusagen zeitgleich) einen Knoten zum Baum hinzu. Für jeden Unterraum führe seperat den Algorithmus aus: gehe mit dem Unterraum zu **(A)**

Und jetzt der Algorithmus in Semi-Code:

Algorithm BUILDKDTREE(P,depth)
Input: A set of points P and the current depth depth.
Output: The root of a kd-tree storing P.

1. **IF** P contains only one point **THEN return** a leaf storing this point
2. **ELSE**
   IF *depth* is even **THEN**
   (a) Split P into two subsets with a vertical line l through the median x-coordinate of the points in P. Let P1 be the set of points to the left and P2 be the set of points to the right. Points exactly on the line belong to P1.
   (b) **ELSE** Split P into two subsets with a horizontal line l through the median y-coordinate of the points in P. Let P1 be the set of points above l and P2 be the points below l. Points exactly on the line belong to P1.
3. V_{right} := ***BUILDKDTREE(P1,depth+1)***
4. V_{left} := ***BUILDKDTREE(P2,depth+1)***
5. Create a node V with Vright and Vleft as its right and left children, respectively.

6. **return** V.

Dieser Algorithmus erzeugt unseren Baum mit einer Zeitkomplexität $O(n \cdot log(n))$ und verbraucht $O(n)$ viel Speicher. Bevor uns schnell klar wird, dass euklidische Räume nicht zwingend nur 2-dimensional sein müssen und wir dem Problem mit Hyperebenen begegnen müssen, sehen wir uns das Beispiel graphisch an.

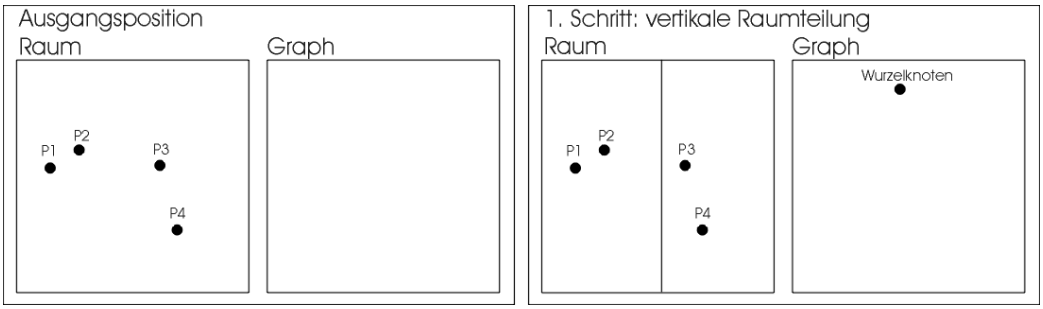

Abbildung 8:

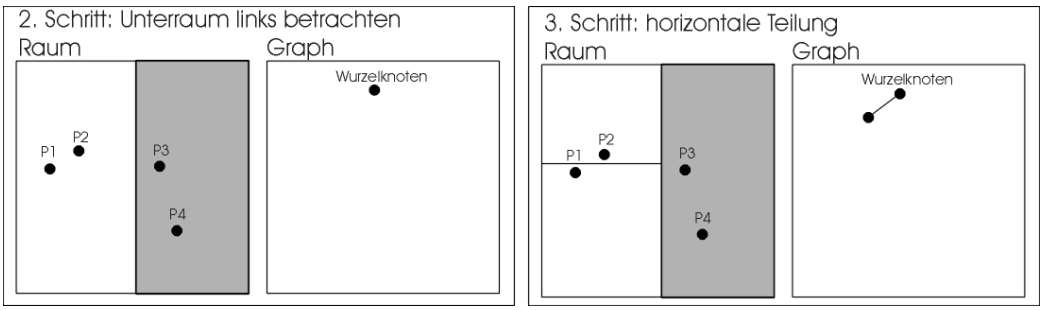

Abbildung 9:

Man beachte, dass die Blätter des so entstandenen Baumes so angeordnet sind, dass Nachbar-Punkte auch auf benachbarte Blätter abgebildet werden. Wichtig für die spätere Suche ist, dass die Knoten noch 2 notwendige Informationen halten:

1. einen Wert d, der angibt bezüglich welcher Dimension der Raum geteilt wurde: mit $d = \{1, ..., k\}$
2. den für genau diese Dimension bestimmten Mittelwert m, der die Stelle kennzeichenet, an der der Raum in zwei Teile gespalten wurde.

Suche im k-d Baum Die Suche nach dem ähnlichsten Punkt im k-d Baum erfolgt in zwei Schritten.

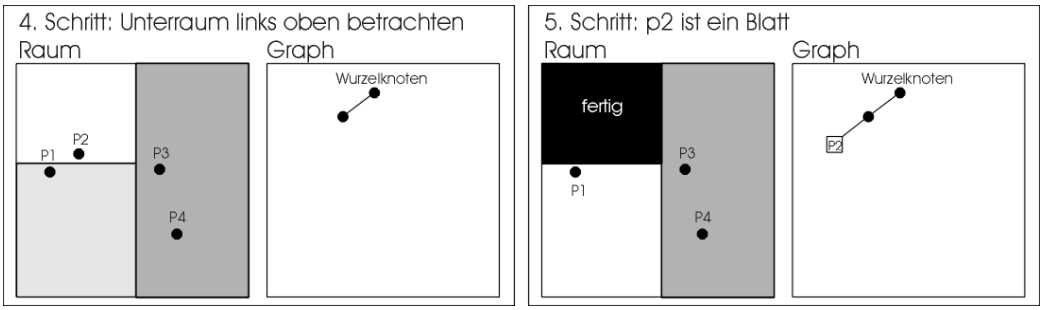

Abbildung 10:

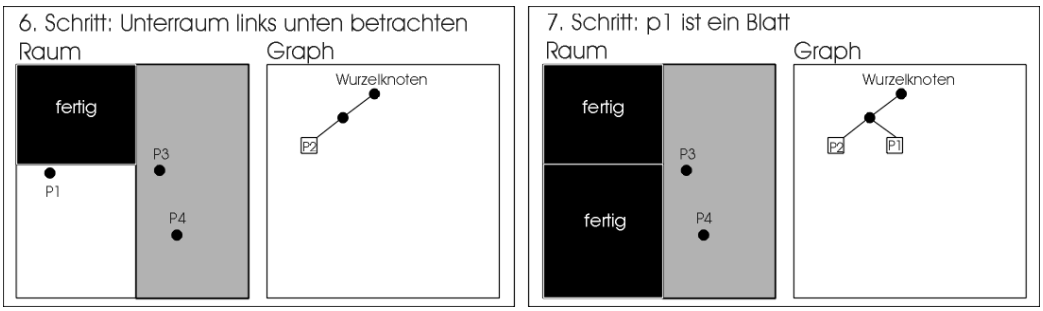

Abbildung 11:

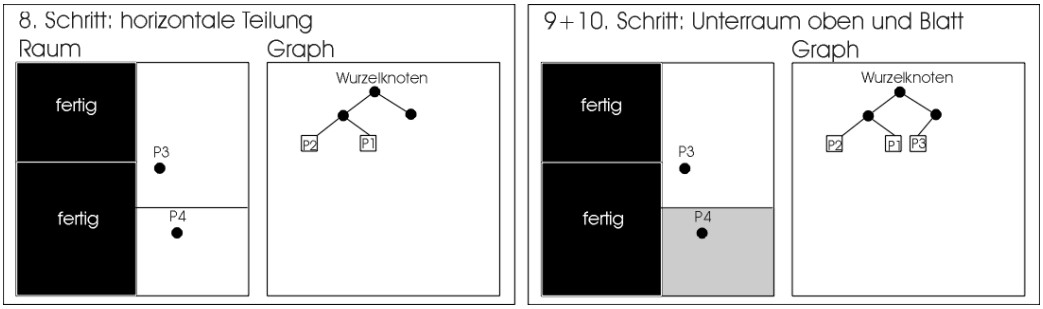

Abbildung 12:

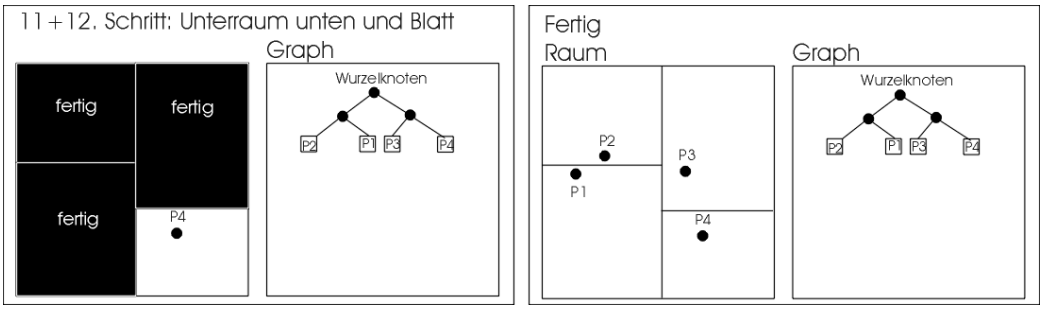

Abbildung 13:

1. nimm den neuen Fall (Punkt) und navigiere[5] von der Wurzel beginnend bis zu einem Blatt = der erste Lösungskandidat
2. Kehre zum letzten durchlaufenen Knoten zurück und prüfe, ob der andere Zweig - den du beim ersten mal nicht gegangen bist - eventuell bessere Lösungen enthält (backtracking). Wiederhole Schritt 2 sinnvoll oft, bis maximal zur Wurzel des Baumes.

5.2.6 M-Tree

Der M-tree ist eine sich rekursiv fortsetzende baumartige Datenstruktur, die einen Metrischen Raum M in Regionen zerlegt, die sich möglicherweise überlappen können.

Voraussetzungen:

(1) Über der Menge der Merkmalsvektoren muss eine Metrik definiert sein.

(2) Der Hüllradius eines Referenz-Objektes O_r erfüllt folgende Ungleichung: $d(O_j, O_r) \leqq r(O_r)$, für alle Objekte O_j des Hüllbaums von O_r.

5.2.7 Die Datenstrukturen

- ein M-tree besteht aus inneren Knoten und Blattknoten
- jeder Knoten enthält eine Menge von höchstens M Einträgen (Kapazität eines Knotens)
- wird die Kapazität berschritten, muss sich der Knoten teilen (Split)
- jeder Eintrag eines Blattknotens ist ein Objekt
- die Einträge der inneren Knoten repräsentieren nur Referenzobjekte

[5] Nutze dazu das in jedem Knoten vorhandene Tupel (d, m)

Mathematische Beschreibung:

- **Eintrag eines Blattknotens:**

$$\boxed{entry(O_j) = [O_j, oid(O_j), d(O_j, P(O_j))]}$$

O_j	Merkmalsbeschreibung des Objekts
$oid(O_j)$	Objektbezeichner
$d(O_j, P(O_j))$	Distanz zwischen O_j und dem Referenz-Objekt$P(O_j)$ (auch parent object) vonO_j

- **Eintrag eines inneren Knotens:**

$$\boxed{entry(O_r) = [O_r, ptr(T(O_r)), r(O_r), d(O_r, P(O_r))]}$$

$ptr(T(O_r))$	Verweis auf den sog. Hüllbaum $T(Or)$von O_r ($T(O_r)$ ist der Knoten, dessen Referenzobjekt O_r ist)
$r(O_r)$	Hüllradius, $r(O_r) > 0$

- jeder Eintrag eines inneren Knotens ist damit das bereits oben erwähnte Referenz-Objekt des ihm zugeordneten Hüllbaums, d.h. $O_r \equiv P(O_j)$

5.2.8 Neue Daten einpflegen – der M-Tree wächst

Einfügen von neuen Objekten:

- der Insert-Algorithmus bewegt sich rekursiv durch den Baum, um den passendsten **Blatt**knoten für ein neues Objekt O_{neu} zu finden
- befinden sich im aktuellen Knoten einige Einträge O_r, die O_{neu} einhüllen, dann wähle ein O_r mit minimalen $d(O_r, O_{neu})$
- gibt es keinen solchen Eintrag, muss der Hüllradius eines Knotens vergrößert werden, wobei die Vergrößerung minimiert werden sollte, d.h. $(d(O_r, O_{neu}) - r(O_r)) \longrightarrow_{min}$
- der M-tree wächst von unten nach oben (bottom-up)
- wird die Kapazität des Knoten N überschritten, wird auf gleicher Ebene ein neuer Knoten N' erzeugt
- mittels der Methode *Promote* werden zwei neue Referenzobjekte ausgewählt

- auf beide Knoten werden nun die (M + 1) Einträge des übergelaufenden Knoten verteilt (die genaue Verteilungstrategie legt der Algorithmus *Partition* fest)
- die Änderungen werden an den darüberliegenden Elternknoten propagiert, der nun ein weiteres Referenzobjekt aufnehmen und sich eventuell selbst teilen muss
- ist der zu partitionierende Knoten der Wurzelknoten, muss ein neuer Wurzelknoten angelegt werden der Baum wächst um eine Etage
- Wichtig! Eigenschaft (2) muss stets eingehalten werden, d.h. alle Objekte des Hüllbaums dürfen von ihrem Referenzobjekt nicht weiter entfernt sein, als es der Hüllradius angibt

5.2.9 Ähnlichkeitssuche im M-tree

Algorithmen zur Ähnlichkeitssuche in M-Trees zielen darauf ab, die Anzahl der Abstandsberechnungen zu minimieren. Deshalb werden alle im M-tree gespeicherten Distanzen, also $d(O_r, P(O_r))$ und $r(O_r)$, ausgenutzt.

Bereichsanfragen (range queries)
Sei Q die Merkmalsbeschreibung des Suchobjektes und $r(Q)$ der Suchradius. Die Anfrage $range(Q, r(Q))$ liefert all die Objekte O_j, für die gilt:

$$d(O_j, Q) \leqq r(Q)$$

.

5.2.10 Zusammenfassung

- der M-tree unterstützt die inhaltsbezogene Ähnlichkeitssuche
- die Ausführung solcher Anfragen ist opimiert auf die Reduzierung der Anzahl der Dinstanzberechnungen
- eignet sich für hochdimensionale Merkmalsvektoren und macht ihn deshalb interessant für die Anwendung auf multimedialen Objekten

6 Weitere Ähnlichkeitsverfahren

6.1 Strukturelle Ähnlichkeit auf Sequenzen

Zur Bestimmung der Ähnlichkeit von Sequenzen werden elementare Umformungsregeln r_i verwendet, denen jeweils Kosten c_i zugeordnet sind. Die Summe über alle Kosten, die bei Regelanwendung zur Überführung einer Sequenz in eine andere aufkommen ist das Ähnlichkeitsmaß. Beim Vergleich von Sequenzen stehen uns im Allgemeinen 3 elementare Umformungen zur Verfügung:

- Löschen eines Elements $\omega(a, \epsilon)$
- Einfügen eines Elements $\omega(\epsilon, b)$
- Ersetzen eines Elements $\omega(a, b)$

6.1.1 Levenshtein Distanz

Die Levenshtein Distanz ist ein Spezialfall eines Distanzmaßes für Sequenzen. Hier wird $\omega(a, \epsilon) = 1$, $\omega(\epsilon, b) = 1$, sowie $\omega(a, b) = 1$ angenommen. Schauen wir uns den Algorithmus doch einmal anhand dieses Distanzmaßes an.
Seien nun also x und y Sequenzen der Länge n bzw. m. Der Präfix von x bis zur Position i werde mit $x[i]$ bezeichnet. Der Abstand zweier Präfixe $x[i]$ und $y[j]$ werde mit $d[i, j]$ bezeichnet. Weiterhin benötigen wir eine Funktion, welche den nächsten Wert für unsere spätere Matrix berechnet.

$$g(i, j) = min(d[i-1, j] + 1, d[i, j-1] + 1, d[i-1, j-1] + cost)$$

Der Parameter *cost* wird im Algorithmus näher erläutert! Nun baue man sich eine $(n+1) \times (m+1)$ Matrix auf und folge dem Algorithmus:

```
Initialisiere die erste Zeile mit 0...n
Initialisiere die erste Spalte mit 0...m
for i = 1 to n
    for j = 1 to m
        if x[i] = y [j] then cost = 0
        else cost = 1
        d[i,j] = g(i,j)
    Next
Next
```

Die Lösung (Levenshtein Distanz) befindet sich nach Abschluss des Algorithmus in der Zelle (n, m) der Matrix.
Siehe auch: http://www.merriampark.com/ld.htm

6.2 Strukturelle Ähnlichkeit

... ist ein auf der Graphentheorie basierendes Ähnlichkeitsverfahren welches besonders die Struktureigenschaften der darzustellenden Objekte erhält und Ableitungen (Inferenzen) darauf ermöglicht.

> *Will man besipielsweise seine Wohnung neu einrichten und dabei die Gestaltung eines Mustter-Raumes aus dem Katalog mit den Möglichkeiten der eigenen vier Wände vergleichen, so muss man zunächst die Objekte (Möbel) des Musterraumes in ihrer Lage zueinander abbilden (Struktur). Dies kann man bewirken, indem man den Knoten eines Graphen die Gegenstände zuzordnet und den Kanten die Beziehungen jener zueinander. Sowahl Knoten als auch Kanten können mehrere Eigenschaftsattribute besitzen.*
> *Ähnlich sind nun zwei Räume, wenn eine hinreichend große Strukturgleichheit (Isomorphie genannt) gibt. Also der Schrank und der Tisch mit der Sitzecke passen, aber der Kaminhocker ist findet keinen Platz. Genau genommen muss man in einem solchen Fall von einer Teil-Isomorphie sprechen - der Kaminhocker ist Schuld.*

Nun wollen wir auch die Attribute für Knoten bzw. Kanten mit in die Betrachtung aufnehmen. Nennen wir den Graphen G und die Menge seiner Knoten K dann bleibt für die Menge der Kanten der Buchstabe E. Alle Attribute, die wir den einem Knoten zuordnen seien der Form $\alpha = (\alpha_{k1}, \ldots, \alpha_{k,a})$ und die Attributre der Kanten nennen wir $\beta = (\beta_{e1}, \ldots, \beta_{eb})$. Ok. Gebe man uns

zwei Graphen. Ziel ist es nun aus dem einen der beiden einen Teilbaum zu extrahieren, der zu einem weiteren Teilbaum des anderen Graphen isomorph ist. Das klingt trivial, nicht war? Ist es aber nicht. Wie programmiert man so etwas? Die Idee ist, die Schrittweise Umformung des einen Baumes in den anderen. Jeder Umformungsschritt kostet etwas. Je höher die Kosten werden, um so unähnlicher werden/sind sich die Bäume.

Folgende Umformungsoperationen sind dazu vorgesehen:

- Lösche einen Knoten
- Ersetze einen Knoten
- Lösche eine Kante
- Ersetze eine Kante

Um auf das Beispiel mit dem Zimmer zurückzukommen. Es sind also aus dem einen Zimmer Möbel zu entfernen, hinzuzufügen und umzustellen, solange, bis beide Zimmer gleich sind. Adäquat verfährt man mit den Attributen (Mein Gott, was sind schon Attribute.).

6.2.1 Eigenschaften und Bewertung

Das Verfahren ist:

- symmetrisch
- transitiv
- idempotent
- Keine Metrik, da es nicht auf einem Distanzmaß basiert
- sollte nur als lokales Ähnlichkeitsmaß eingesetzt werden, da es bei größeren Graphen ein Zeitproblem geben wird

6.3 Ähnlichkeiten zwischen Gestalten

Idee ist es, in Graphiken bestimte Formen/Gestalten wie einen Kreis, ein Quadrat u.s.w. zu finden und auf dieser Basis zwei Graphiken bezüglich ihrer Änlichkeit zu bewerten. Diese speziellen Formen sollen benutzt werden, um eine neue Graphik nach Analyse einen Index zuzuordnen, mit dessen Hilfe später der Fall (Graphik) schnell aufgefunden werden kann. Zunächst aber

brauchen wir eine Referenzmenge für jede unseren Formen[6], welche wir dann in den Graphiken suchen werden.

Möglichkeiten um Gestalten in Fällen zu finden:

- man könnte für jede Gestalt einen eigenen Algorithmus entwickeln, um diese zu finden
- man benutzt eine abstrakte Darstellungsform (Skizze), in der sowohl die Fälle als auch die Referenzen vorliegen

6.3.1 Referenzmenge, Gestalten und Skizzen

Jede Gestalt oder Form hat nun einen Menge voller Skizzen, Referenzmenge genannt, die sie, die Form, beschreiben. Ziel ist es nun, Skizzen zu finden, die für ähnliche Dinge gleich und für nicht ähnliche verschieden sind.

6.3.2 Gestalt mit Skizzen vergleichen

Folgende Schritte sind vonnöten:

- **Referenzmenge erzeugen**
- **umschließender Ramen**: um die betrachteten Objektgruppen wird ein Ramen gesetzt
- **Gitter**: in dem Ramen wird ein 3·3 Gitter platziert, um von Skalierung und Verzerrung zu abstarahieren
- **Ausrichtung**: jedes Objekt wird als eine Linie mit entsprechender Ausrichtung skizziert
- **Inner merge**: gleich ausgerichtete Objekte innerhalb eines Gitterfeldes werden zu einem Element verschmolzen
- Objekte **ohne Ausrichtung**: gibt es nur Objekte ohne gemeinsame Ausrichtung, so wird versucht über Mittelwerte eine Gesamtausrichtungfür die ganze Gruppe zu errechen

[6]Einen Sack mit verschiedenen Quadraten, Dreiecken und Kreisen, die proportioniert und zuammengesetzt eine bestimmte Form ergeben.

- **erster Eintrag** in die Referenzmenge: Elemente in einem Gitterfeld werden zentriert und dann als Skizze in die entprechende Menge eingetragen
- **Outer merge:** es wird versucht, die Regel des Inner merge auf Elemente aus benachbarten Gitterfeldern anzuwenden.
- **zweiter Eintrag** in die Referenzmenge: wieder werden alle Elemente in den Gitterfeldern zentriert und die entsprechende Skizze in die jeweilige Referenzmenge eingetragen

Nun muss die Fallbasis mit Indizies versehen werden. Hierzu wird der aktuelle Fall mit dem ersten Sack der ersten Gestalt appliziert.

- die Objektgruppe wird skizziert
- nach einem Inner merge wird in den Referenzmengen nach einer gleichen Skizze gesucht
  - wird diese gefunden, wird der Name der Skizze zum Teil des Indizies
  - wird keine Skizze gefunden, wird nach dem Outer merge nochmals nach Skizzen gesucht
- wird keine Übereinstimmung gefunden, kann man eventuell von ein beschädigten Gestalt ausgehen und versuchen diese zu reparieren
- wenn immernoch kein Ergebnis vorliegt, wird der Fall schließlich in Bezug auf jede Skizze Modifiziert, bis er dieser entspricht; dabei werden die Schritte mit einer Kostenfunktion bewertet. Wurden schließlich alle Skizzen „über den Fall gepresst“ nimmt man die Skizze(n) mit der höchsten Ähnlichkeit, also die, mit den geringsten Umformungskosten.

6.3.3 Eigenschaften und Bewertung

- die Indizies sind invariant gegen Verzerrung, Rotation, Spiegelung, Translation und der Anzahl der beteiligten Objekte
- jedoch nicht gegen Verschiebung der Objekte im fokusierten Gebiet
- die Ähnlichkeit ist nicht transitiv und auch keine Metrik
- kann für große Fallbasen in form eines Assoziativspeichers eingesetzt werden

6.4 Quasi-analoge Ähnlichkeit

Dieses Verfahren[7] wurde speziell für die Ähnlichkeitsanalyse digitaler Bilder entwickelt, welche als Bitmaps vorliegen. Die Idee ist, die Bildauflösung (Pixel pro Inch) stufenweise zu manipulieren und zu vergleichen. Gegeben sei eine Menge von Referenzbildern[8], die zunächst in z.B. 10 verschiedene Auflösungen transformiert werden. Die so gewonnenen Bilder werden Knoten eines Baumes zugeordnet; der Gestalt, dass die scharfen[9] Bilder die Blätter darstellen, während immer zwei Knoten benachbarter Ebenen genau dann miteinander verbunden sind, wenn sie das selbe Bild repräsentieren. Der Baum wird also zum Wurzelknoten hin immer unschärfer und ist letzlich homogen grau in der Wurzel.
Soll nun ein neues Bild auf Ähnlichkeit zu den schon vorhandenen hin überprüft werden, wird es ebenfalls in die unterschiedlichen Schärfegrade umgeformt. Nun wird es mit der niedligsten Aufläsung mit allen Referenzbildern ebendieser Auflösung, die im Baum auf einer Ebene angeordnet sind, auf Gleichheit überprüft. Alle Knoten, die das Gleichheitskriterium erfüllen, werden für den nächsten Suchschritt gespeichert. Jetzt wird der Schärfegrad um eins erhöht und auf der nächsten Ebene des Baumes verglichen u.s.w.. Sind wir an der untersten Ebene, der Blattebene, angelangt, berechnen wir die noch abweichende Pixelzahl unseres neuen Bildes zu den relevanten Blattbildern und wählen das mit der geringsten Abweichung als das Ähnlichste.

6.4.1 Eigenschaften und Bewertung

- symmetrisch
- nicht transitiv
- keine Metrik
- um Rotations- und Reflekionsinvarianz zu realisieren, müssen alle acht durch Drehung und und Spiegelung entsehenden Varianten in die Fallbasis aufgenommen werden
- liefert keine Anhaltspunkte für eine spätere Adaption

[7] Auch - „Watching from distance“ - Ähnlichkeit genannt.

[8] Eine Menge von Bildern also, mit denen wir andere, neu hinzugekommene auf Ähnlichkeit vergleichen wollen.

[9] Mit der höchsten Auflösung.

Zudem kann nicht garantiert werden, dass immer die wirklich ähnlichsten Bilder gefunden werden, da bei der Skalierung der Schwerpunkt als Mittelpunkt zentriert wird. Durch eine recht gute Zeitkomplexität eignet es sich besonders zur Vorauswahl.