Problem Solving by

Case-Based Reasoning

PART 1

Machine Learning

Sommersemester 2009

13.05.2009

Thomas Gabel

Machine Learning Lab

Agenda

1. INTRODUCTION

What is Case-Based Reasoning?

Case-Based Reasoning is …

- an approach to model the way humans think
- an approach to build intelligent systems

Central Ideas:

- store experiences made ➔ as cases
- solving a new problem do the following
  - recall similar experiences (made in the past) from memory
  - reuse that experience in the context of the new situation (reuse it partially, completely or modified)
  - new experience obtained this way is stored to memory again

Classification of CBR

- sub-discipline of Artificial Intelligence
- belongs to Machine Learning methods
  - learning process is based on analogy (not on deduction or induction)
  - best classified as supervised learning
- one of the few commercially/industrially really successful AI methods
  - customer support, help-desk systems: diagnosis and therapy of customer's problems, medial diagnosis
  - product recommendation and configuration: e-commerce
  - textual CBR: text classification, judicial applications (in particular in the countries where common law (not civil law) is applied [like USA, UK, India, Australia, many others])
- applicability also in ill-structured and bad understood application domains

Case-Based Reasoning and Cases

- „Case-Based Reasoning is […] reasoning by remembering." Leake, 1996
- „A case-based reasoner solves new problems by adapting solutions that were used to solve old problems." Riesbeck & Schank, 1989
- „Case-Based Reasoning is a recent approach to problem solving and learning […]" Aamodt & Plaza, 1994
- „Case-Based Reasoning is both […] the ways people use cases to solve problems and the ways we can make machines use them." Kolodner, 1993

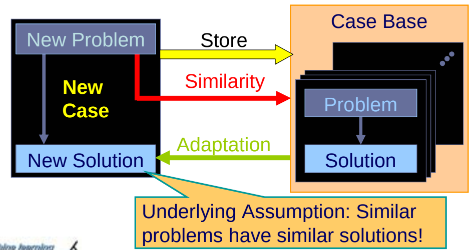
What is a Case?

a) Cognitive Science View:
Cases are abstractions of events, that are limited in space and time; they represent episodic knowledge.

b) Technical View:
A case is a desciption of a problem situation (that actually occurred) together with certain experiences that could be obtained during processing and solving the problem.

Simplified CBR Model

Solve new problems by selecting cases used for similar problems and by (eventually) adapting the belonging solution.



When is CBR of Relevance?

- When a domain theory does not exist, but example cases are easy to find.
- When an expert in the domain is not available, is too expensive, or is incapable of articulating verbally his performance, but example cases are easy to find.
- When it is difficult to specify domain rules, but example cases are easy to find.
- When cases with similar solutions have similar problem descriptions.
  - i.e. there exists a similarity metric for problem descriptions and a corresponding set of adaptation rules
- When a case base already exists.

Contents of a Case

Mandatory
- problem part
- solution part

Optionally
- context (e.g. justifications)
- pointer to other relevant cases
- solution quality assessment
- steps of the solution

➔ The main difficulty arises, when the actual situation is not identical to the previous one. Then, inexactness is involved.
➔ A main feature of CBR techniques is that they allow inexact / approximate reasoning in a controlled manner.

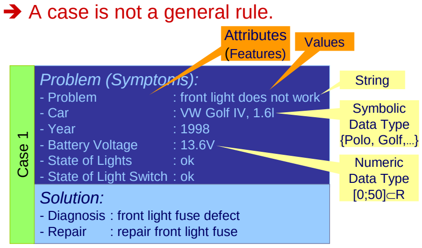
A Simple Example Scenario: Call Center (I)

- Task: Technical Diagnosis of Car Faults
  - symptoms are observed (e.g. engine does not start) and values are measured (e.g. battery voltage = 6.3V)
  - goal: find the cause for the failure (e.g. battery empty) and a repair strategy (e.g. change battery)
- Case-Based Diagnosis
  - a case describes a diagnostic situation and contains
    - a description of the symptoms
    - description of the failure and the cause
    - description of a repair strategy
  - store a collection of cases in the case base
  - find a case similar to the current problem and reuse the repair strategy

A Simple Example Scenario: An Example Case (II)

- A case describes a particular diagnostic situation.
- A case records several features and their specific values occurred in that situation.

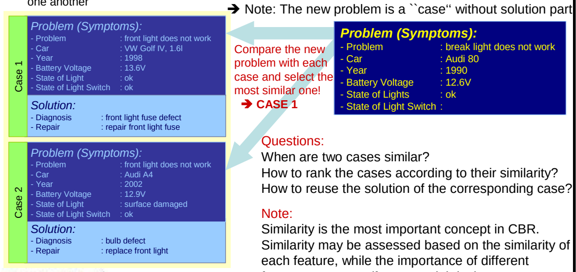
  ➔ A case is not a general rule.



A Simple Example Scenario: Solving a New Diagnostic Problem (III)

Case Base with Two Cases
➔ each case describes one particular situation
➔ all cases are independent of one another

A New Problem (Query) Has to Be Solved
➔ we make several observations in the current situation
➔ observations define a new problem
➔ not all attribute values have to be known
➔ Note: The new problem is a ``case" without solution part

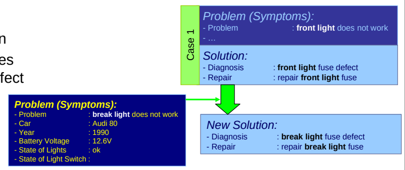


Questions:
When are two cases similar?
How to rank the cases according to their similarity?
How to reuse the solution of the corresponding case?

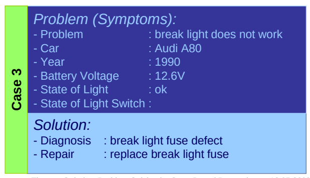
Note:
Similarity is the most important concept in CBR.
Similarity may be assessed based on the similarity of each feature, while the importance of different features may vary (feature weighting).

A Simple Example Scenario: Reuse and Retain (IV)

- Reuse
  - adapt the solution
  - how do differences in the problem affect the solution

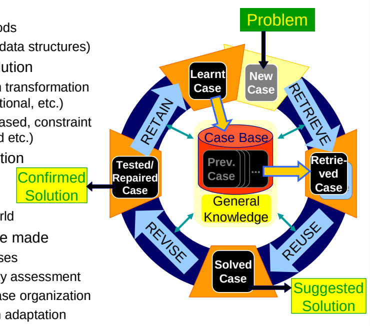


- Retain
  - if diagnosis is correct: store new case
  - add case to case base



CBR Cycle (R4, [Aamodt&Plaza, 1994])

- retrieve: find most similar case(s)
  - similarity measures
  - explanation-based methods
  - case-base organisation (data structures)
- reuse: transform/adapt solution
  - different types of solution transformation (none, interactive, derivational, etc.)
  - different methods (rule-based, constraint satisfaction, model-based etc.)
- revise: verify/improve solution
  - no verification
  - verification by simulation
  - verification in the real world
- retain: keep the experience made
  - learn new cases
  - learn similarity assessment
  - learn case base organization
  - learn solution adaptation



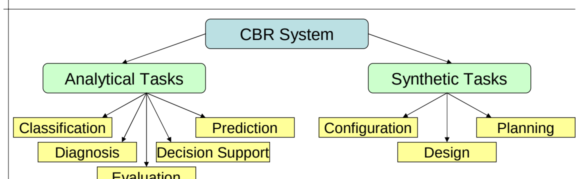
Advantages of CBR (I)

- Avoidance of High Knowledge Acquisition Effort
  - case knowledge is usually easily acquirable
  - not much general knowledge required
- Simpler Maintenance of the Knowledge in the System
  - maintenance by adding/removing cases from the case base
  - cases are independent of one another and easily interpretable (even for non-experts)
- Facilitation of Intelligent Retrieval (compared to data-base systems)
  - DBMS often give too few/many results

Advantages of CBR (II)

- High Quality of Solutions for Poorly Understood Domains
  - case-based systems can be made to retain only ``good" experience in memory
  - if only little adaptation is necessary for reuse, this will not impair the solution's quality too much
- High User Acceptance
  - provided solution corresponds to actual experience
    → may increase trust in the solution
  - selected case and solution adaptation can be explicitly presented to the user
  - problems of rule-based / neural network-based systems
    - black boxes
    - inference process is not traceable or hidden
    - provided solutions are difficult to explain

Typical Application Fields (I)



- Remarks concerning synthetic tasks:
  - main focus is on composing a complex solution from simpler components
    → focus is often on solution adaptation
  - configuration: e-commerce scenario → product configuration (e.g. personal computers)
  - design: reuse of construction plans in civil engineering
  - planning: production planning

Typical Application Fields (II)

- Remarks concerning analytical tasks:
  - main focus is on analysing a given situation
  - classification (assign objects to a class $K_i \in \{K_1, \ldots, K_n\}$)
    → e.g. recognition of sponges
  - diagnosis (classification + verification + therapy)
    → e.g. fault diagnosis in Airbus engines
  - evaluation/regression (like classification, but assignment of real-valued assessments):
    → e.g. credit risk assessment
  - decision support (search for specific information relevant for decision-making)
    → e.g. web-based product catalogues, like online travel agencies
  - prediction (like classification + time dependency)
    → e.g. prediction of the probability of failure of a machine's part

CBR for Classification

- A classifier for a set M is a mapping f:M→I (where I is a finite index set).
  - ➔ A case-based classifier is given by a case base, a similarity measure and the principle of the nearest neighbour.
- **Definition:** Given a case base CB, a similarity measure sim and an object (problem) q∈M, we call c=(p,s)∈CB the *Nearest Neighbour* to q, if: for all (p‘,s‘)∈CB it holds sim(q,p)≥sim(q,p‘).
- NN-Classification: Each new object (query) q∈M is assigned the class s∈I of q‘s nearest neighbour in CB.
- Note: The classifier defined by the pair (CB,sim) is not unique, if there is more than one nearest neighbour.
- Extension to kNN-Classification: The k most similar neighbours of q are considered. Typically, a majority voting is applied to determine the class of the query q.

k-Nearest Neighbor Classification

- Demonstration video on the k-NN classifier

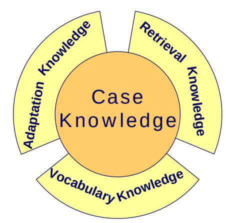
(c) Antal van den Bosch, Tilburg University

2. KNOWLEDGE AND CASE REPRESENTATION

What forms of knowledge are parts of a CBR system?
How can cases be represented?

Knowledge Container Model [Richter, 1989]

- „In order to solve problems, one needs knowledge.“
- Knowledge of a CBR System
  - vocabulary: knowledge representation
  - retrieval: similarity assessment (measures)
  - solution transformation: rules
  - cases
- Knowledge Management
  - as the environment may change, maintenance of the containers‘ contents over the lifetime of the CBR system is crucial to guarantee its continued usability



Case Contents

Problem / Situation Information

➔ must cover all the information that is necessary to decide if this case is applicable for a new situation
- target of the problem
- constraints
- characteristics

➔ new situation = query

Solution

➔ contains all the information that is describes a solution to the problem sufficiently accurately
- ➔ solution itself
- ➔ justifications
- ➔ possible alternative solutions
- ➔ steps that were tried, but failed

Solution Evaluation

➔ feedback from the real world
- ➔ How good was the solution for the problem?

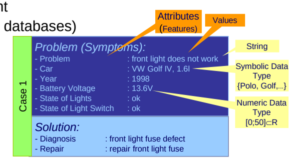
Case Representation Formalisms (I)

Attribute-Value Based Case Representation

- Case (problem and solution) is represented by pairs of attributed and belonging values.
  - e.g.: price = 9.95€
- Set of attributes $\{A_1,\dots,A_n\}$ is (in general) fixed for all cases.
- To each attribute A_i there is an associated domain D_i and for each attribute‘s value it holds $a_i \in D_i$, e.g.
  - numerical attributes (Integer or Real or subsets of those)
  - symbolic attributes (finite domains, $D_i=\{d_1,\dots,d_m\}$
  - textual attributes (strings)
- Note:
  - Choice of attributes and corresponding domains to represent cases represents general knowledge: vocabulary knowledge.
  - Choice of domains is mainly influenced by the requirements for similarity computation and solution adaptation.

Case Representation Formalisms (II)

- Choice of attributes
  - must allow for the decision whether a case and a new situation are similar
  - should avoid redundancies
  - should represent independet properties of a case
- Disadvantages
  - no structural or relational information is representable
  - no ordering information (e.g. sequence of actions) is representable
- Advantages
  - straightforward representation
  - easy to understand and implement
  - cases are easy to store (usage of databases)
  - efficient retrieval
- Example
  - recall the example from the Introduction



Case Representation Formalisms (III)

Object-Oriented Case Representation

- Refinement and more structured extension of attribute-value based representation
- Compositing of related attributes to object descriptions; each object is described by a fixed set of attributes ➔ Case = Set of Objects

Graph- and Tree-Based Representation

- e.g. suited for atomic/molecule structures or electrical circuit designs

First-Order-Based Case Representation

- problems and solutions are represented as sets of Grundatome (variable-free)

Hierarchical Case Representation

- each case is represented on several levels of abstraction

Generalised Cases

- each case describes sets of cases at once, which are highly similar to one another ➔ smaller case bases, simplified case/solution adaptation

3. SIMILARITY

When is a new problem (query) similar to a case's problem part?
What forms of similarity measures are suitable?

Meaning of Similarity

- Similarity is the central notion in Case-Based Reasoning.
- Similarity is always considered between problems (not solutions of cases).
- Selection of cases during the ``Retrieve" phase is based on the similarity of cases to a given query.

➔ Observation I: There is no universal similarity; similarity always relates to a certain purpose.
  - e.g. two cars can be similar if they have the same max speed or cost approximately the same → different aspects of similarity

➔ Observation II: Similarity is not necessarily transitive.
  - e.g. 10€ are similar to 12€, 12€ are similar to 14€ … 100€ are similar to 102€. *But:* 10€ are not similar to 102€ → property of ``small numeric difference" is intransitive

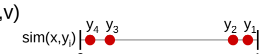
➔ Observation III: Similarity does not have to be symmetric.

Similarity and Utility

- Purpose of Similarity: Selection of solutions that can be easily transferred / adapted to the problem at hand.
- Similarity = Utility for Solving a (new) Problem
- *Note:*
  - Utility is an a-posteriori criterion: In general, the utilitiy (of a case) can be estimated after having solved the problem.
  - Similarity concerning problem situations is an a-priori criterion: Similarity must be estimated before solving the problem.
- **Goal:** Similarity must approximate utility as accurately as possible.

Similarity Measures

- *Idee:* Numerical modelling of similarity, capturing the degree of similarity
- **Definition:** A ***Similarity Measure*** on a set M is a real-valued function sim: $M^2 \rightarrow [0,1]$.
  We say that sim is
  - reflexive iff. $\forall x \in M$: sim(x,x) = 1
  - symmetric iff. $\forall x,y \in M$: sim(x,y) = sim(y,x)
- Beyond ordinal information, similarity measures allow for a quantitative statement on the degree of similarity.
- **Definition:** Each similarity measure induces a ***similarity relation*** R_{sim} as
  $R_{sim}(x,y,u,v)$ iff. $sim(x,y) \geq sim(u,v)$
  $y \geq_z x$ iff. $sim(z,y) \geq sim(z,x)$



Distance Measures

- **Definition:** A ***Distance Measure*** on a set M is a real-valued function d: $M^2 \rightarrow \Re^+_0$.
  We say that sim is
  - reflexive iff. $\forall x \in M: d(x,x) = 0$
  - symmetric iff. $\forall x,y \in M: d(x,y) = d(y,x)$
- **Definition:** A distance measure d on a set M is a ***Metric*** and (M,d) a ***Metric Space*** if
  - $\forall x,y \in M$: $d(x,y) = 0 \Rightarrow x=y$
  - $\forall x,y,z \in M$: $d(x,y) + d(y,z) \geq d(x,z)$
- **Definition:** Each distance induces a ***similarity relation*** R_d as
  $\forall x,y,u,v \in M$: $R_d(x,y,u,v)$ iff. $d(x,y) \leq d(u,v)$
  $\forall x,y,z \in M$: $y \geq_z x$ iff. $d(z,x) \leq d(z,y)$

Relation Between Distance and Similarity Measures

- **Definition:** A similarity measure sim and a distance measure d are called ***Compatible*** if and only if
  $\forall x,y,u,v \in M: R_{sim}(x,y,u,v) \leftrightarrow R_d(x,y,u,v)$
- ***Lemma** (Measure Transformation):* If there is a bijective, order-reversing mapping $f: \Re^+_0 \rightarrow [0,1]$ with
  $f(0) = 1$
  $f(d(x,y)) = sim(y,x)$
  then sim and d are compatible.
- *Note:* A transformation function f can be employed to construct a compatible pendant for a given sim or d, respectively.
- *Examples:*
  - $f(x) = 1 – x/(x+1)$
  - $f(x) = 1 – x/x_{max}$

Examplary Similarity Measures (I)

(for attribute-value based case representations)

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

- for binary features
- $x=(x_1,\ldots,x_n)$, $x_i \in \{0,1\}$
- $H(x,y) \in \{0,\ldots,n\}$
- H(x,y) is the number of attributes with differing values
- H is a distance measure: $H(x,x)=0$, $H(x,y)=H(y,x)$
- $H((x_1,\ldots,x_n), (y_1,\ldots,y_n)) = H((1-x_1,\ldots,1-x_n), (1-y_1,\ldots,1-y_n))$

SMC (Simple Matching Coefficient) $SMC(x,y) = 1 - \frac{n-(a+d)}{n} = \frac{a+d}{n} = 1 - \frac{b+c}{n}$

- $a=\Sigma x_i y_i$, $b=\Sigma x_i(1-y_i)$, $c = \Sigma(1-x_i)y_i$, $d = \Sigma(1-x_i)(1-y_i)$
  $\Rightarrow n=a+b+c+d \Rightarrow H(x,y) = b+c = n-(a+d)$
- transformation into a compatible similarity measure by $f(x)=1-x/x_{max}$ yields the simple matching coefficient

Examplary Similarity Measures (II)

(for attribute-value based case representations)

Measures for Real-Valued Attributes

- $x_i, y_i \in \Re$ for all i
- generalisations of the Hamming distance
  - city-block metric d_1
  - Euclidean distance d_2
  - weighted Euclidean distance d_{2w}
  - p-norm d_p
- cf. lectures on *Vector Space & Linea Algebra*

Note: Similarity measures for other case representations (e.g. object-oriented, graph-based, etc.) are not considered in this lecture; see literature references.

Local-Global Principle

- case description by n attributes $A_1,\ldots,A_n$
- each attribute has a certain type T_i (e.g. numeric)
- **Local Similarity**
  - a seperate similarity function is used for each attribute:
    $sim_{Ai}: T_i \times T_i \rightarrow [0,1]$
  - local measures are depending on the respective type T_i of the attribute A_i
- **Global Similarity**
  - $sim(x,y) = sim((x_1,\ldots,x_n),(y_1,\ldots,y_n))$
    $= F(sim_{A1}(x_1, y_1),\ldots, sim_{An}(x_n, y_n))$
  - $F:[0,1]^n \rightarrow [0,1] \Rightarrow$ **Amalgamation Function**
  - requirements on F:
    - F is monotonous in each of its arguments
    - $F(0,\ldots,0) = 0$ and $F(1,\ldots,1) = 1$

Very frequently used in practice!

Examples:
- Weighted Average
$F(s_1,\ldots,s_n)= \sum_{i=1}^{n} w_i s_i$
- Maximum
$F(s_1,\ldots,s_n) = \max\{s_1,\ldots,s_n\}$
- k-Minimum
$F(s_1,\ldots,s_n) = s_{ik}$ with $s_{i1} \leq s_{i2} \leq \ldots \leq s_{in}$
- etc.

Local Similarity Measures (I)

(for unordered symbolic and integer/real-valued attribute types)

Similarity Tables

- for attributes with symbolic type $T_A=\{v_1,\ldots,v_k\}$
- sim table/matrix $sim_A(x,y)=s[x,y]$
- example: attribute ``RAM-Type" with $T_A=\{SD,DDR,RD\}$

q \ c	SD	DDR	RD
SD	1.0	0.9	0.75
DDR	0.5	1.0	0.75
RD	0.25	0.5	1.0

RAM-Type

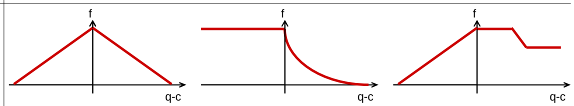
- reflexive similarity measure: diagonal elements $s[k,k]=1$
- symmetric similarity measure: $s=s^T$

Difference-Based Similarity Functions

- for attributes with numeric type (e.g. integer or real-valued)
- similarity is based on the numerical difference between case and query value
  - linearly scaled domains: $sim_A(x,y)=f(x-y)$
  - exponentially scaled domains: $sim_A(x,y)=f(\log(x)-\log(y))$
- typical requirements on f
  - $f:\Re \rightarrow [0,1]$ or $Z \rightarrow [0,1]$
  - $f(0)=1$ (reflexivity)
  - f(x) is monotonously falling/incresing
- examples: next slide

Local Similarity Measures (II)

(for unordered symbolic and integer/real-valued attribute types)



- symmetric
- simple distance

- asymmetric
- x = query q, y = case c
- query is minimal requirement
- e.g. minimal horse power of a car required by a customer

- asymmetric
- x = query q, y = case c
- query depicts maximal value
- e.g. maximal price of a product
- background knowledge included (for q-c>0)

Local Similarity for Other Types ➔ *not considered here*

- ordered symbolic data types
  - e.g. T_A={small,average,tall}
- taxonomic data types
  - elements of T_A can be arranged within a taxonomical (tree) structure
  - e.g. attribute to describe the types of CPUs

4. SIMILARITY-BASED RETRIEVAL

How to retrieve a query's nearest neighbour(s)?

Sequential Retrieval

- Retrieval Task
  - Input
    - case base CB={c_1,…,c_n}
    - similarity measure sim
    - query (new problem) q
  - Output
    1. most similar case c_i
       or
    2. m most similar cases {c_{i1},…,c_{im}}
       or
    3. all cases {c_{i1},…,c_{il}} which have at least a similarity of sim_{min} to q
  - Main Problem: Efficiency
  - Question: How can the case base be organised in such a way to support an efficient retrieval?
- Sequential Retrieval
  - iterates over all c∈CB and calculates sim(c,q)
  - returns the most similar / m most similar cases to q
  - complexity: O(n)
  - Advantages
    - easy to implement
    - no index structures to maintain
    - usability of arbitrary similarity measures
  - Drawbacks
    - problematic for large case bases
    - effort independent of query
    - effort independent of m

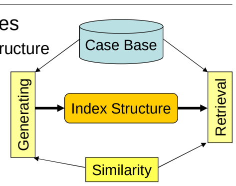
Two-Stage Retrieval

- Idea: **MAC/FAC** (many are called, few are chosen)
  1. preselection of possible solution candidates M_q⊆CB, where M_q = { c∈CB | fac(q,c) }
  2. use sequential retrieval on M_q

  ➔ ***Problem:*** Finding of an adequate predicate fac
- Examples for predicate fac
  - partial equality: fac(q,c) iff. q and c are identical w.r.t. at least one attribute
  - local similarity: fac(q,c) iff. q and c are sufficiently similar w.r.t. to each attribute
  - partial local similarity: fac(q,c) iff. q and c are sufficiently similar w.r.t. to at least one attribute
- Advantage: good performance $|M_q|$ if is small
- Drawbacks
  - retrieval errors may occur → α-error: A case c that is sufficiently similar to q w.r.t. sim is not found (because not considered during preselection).
    ➔ completeness of retrieval is not guaranteed
  - determination of an adequate predicate for preselection is usually difficult

Case Retrieval with kd-Trees

- Index-Oriented Retrieval Procedures
  - preprocessing: generating an index structure
  - retrieval: exploit the index structure to efficiently access the cases



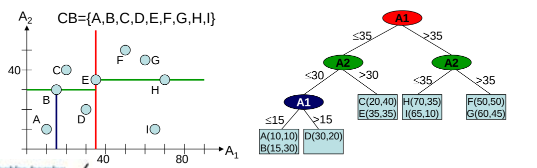
- Possible Index Structur: kd-Tree
  - A ***kd-Tree*** is a k-dimensional binary search tree to support an efficient search over data sets.
  - Idee: partitioning of the data (here: the case base) into small intervals.
  - ordering within a binary tree (similar to a decision tree)
  - during retrieval
    - stepping through the tree from root to the leaves
    - backtracking is possible (unlike in decision trees)

Definition of kd-Trees

- Input
  - k ordered domains T_1,…,T_k for attributes A_1,…A_k
  - case base CB⊆T_1×… ×T_k
  - parameter b (bucket size)
- **Definition:** A *kd-Tree T(CB) for case base CB* is a binary tree, that is defined as
  - if |CB|≤b: T(CB) is a leaf of the tree (called bucket), denoted CB
  - if |CB|>b: T(CB) is a tree whose
    - root is denoted with an attribute A_i and a value $v_i \in T_i$ and
    - two sub-trees $T_{\leq}(CB_{\leq})$ and $T_{>}(CB_{>})$ are kd-trees, too, with
      - $CB_{\leq}$:= { (x_1,…x_k) ∈ CB | $x_i \leq v_i$ } and
      - $CB_{>}$:= { (x_1,…x_k) ∈ CB | $x_i > v_i$ }

Properties of kd-Trees

- kd-tree partitions the case base
  - root represents the entire case base
  - a leaf (bucket) represents a subset of the case base that does not have to be further partitioned
  - at each inner node the case base is partitioned, being divided on the basis of some specific value of an attribute
- Example:



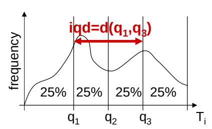
Generating kd-Trees

- Algorithm

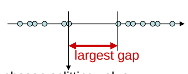
```
PROCEDURE CreateTree(CB): kd-Tree
if |CB|<b
then
  return leaf node marked with case base CB
else
  A_i := choose_attribute(CB)
  v_i := chose_split_value(CB,A_i)
  return
    tree whose root is marked with Ai and vi
    and which has sub-trees
      CreateTree( {(x_1,…,x_k)∈CB | x_i<=v_i} )
      CreateTree( {(x_1,…,x_k)∈CB | x_i>v_i } )
```


Attribute Selection and Splitting Values

- various methods usable for attribute selection
  - entropy-based
  - inter-quartile distance
    ➔ choose the attribute with the biggest inter-quartile distance **iqd**



- determination of splitting values
  - median splitting: choose median as splitting value
  - maximum splitting: search for the ``largest gap"



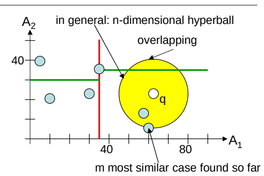
Retrieval Algorithm Using kd-Trees

- Algorithm
  - simqueue is a global data structure holding the m most similar cases as well as their corresponding similarities with respect to q

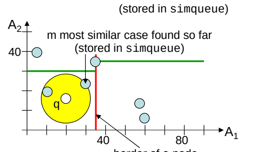
```
PROCEDURE Retrieve(K: kd-Tree, q: Query)
if K is leaf node
then
  forall c∈CB do
    if sim(q,c)> simqueue[m].sim
    then insert c into simqueue
else
  A_i := the attribute K is marked with
  v_i := the splitting value K is marked with
  q_i := the attribute A_i's value of q
  if q_i<=v_i
  then
    Retrieve(K_≤)
    if BOB-Test is fulfilled then Retrieve(K_>)
  else
    Retrieve(K_>)
    if BOB-Test is fulfilled then Retrieve(K_≤)
  if BWB-Test is fulfilled
  then terminate retrieval returning simqueue
  else return
```


BOB- and BWB-Tests

- BOB-Test:
  Can there be – in the neighbouring sub-tree – any more similar cases (to query q) than the m most similar cases already found?



- BWB-Test:
  Is it guaranteed that there is no case in a neighbouring sub-tree which is more similar to the query q than the m-most similar case found so far?



Discussion of kd-Tree Retrieval

- Restriction
  - Retrieval using kd-trees guarantees finding the m nearest neighbours, if the similarity measure used fulfills the following condition:
    Compatibility with ordering and monotony:
    $\forall x_1,\dots,x_n$ and x_i', x_i'' if $x_i <_i x_i' <_i x_i''$
    then $sim((x_1,\dots,x_n), (x_1,\dots,x_i',\dots x_n))$
    $\geq sim((x_1,\dots,x_n), (x_1,\dots,x_i'',\dots,x_n))$
- Advantages
  - efficient retrieval
  - effort depends on the number m of most similar cases to find
  - incremental extension of the kd-tree is possible
- Drawbacks
  - higher costs for building up the index structure (kd-tree)
  - restrictions implied by kd-trees
    - usability for ordered domains only
    - unknown attribute values are difficult to handle
    - only for monotonous similarity measures that are compatible with the ordering of the respective attribute's domain

Other Retrieval Methods

- Several further advanced retrieval approaches
  - high efficiency
  - general usability depends on problem setting (e.g. case modelling)
- Examples
  - Case Retrieval Nets [Burkhardt&Lenz]
  - Retrieval with ``Fish and Shrink“ [Schaaf, 1996]
  - Case Retrieval on Top of Relational Databases Utilising SQL [Schumacher, 2000]

1 2 3 4 5 6 7 8

Part 1 (overed today) ***Part 2: Outlook***

Thanks!

Questions?

tgabel@informatik.uni-freiburg.de