Machine Learning

Michael M. Richter

Case Based Reasoning

Email: mrichter@ucalgary.ca
Learning by Experience

• Experiences are stored stories of
  – successful solutions: Do it again!
  – or failures: Avoid this!

• These stories are in a memory.

• We have two problems to use them:
  – The situation in the experience is not exactly the same as the actual solution
  – How to find the most useful experience?

• For the first problem we observe that the experience is not totally different from the actual situation; it is similar or analogous.

• Experiences are also called cases.
Case Based Reasoning

• Case Based Reasoning makes systematic use of these ideas.

• Central is the concept of similarity: It determines when an old experience is possibly useful for solving a new problem.

• The success depends on:
  – The set of available cases (the case base)
  – The quality of the similarity measure.

• This approach has been generalized to situation where no experiences are available by establishing a similarity between the problem and the solution directly.
Description of an Experience (1)

- We deal with a particular diagnostic situation
- The story records several features and their specific values occurred in that situation.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem (Symptoms)</td>
<td></td>
</tr>
<tr>
<td>• Problem: Front light doesn’t work</td>
<td></td>
</tr>
<tr>
<td>• Car: VW Golf IV, 1.6 l</td>
<td></td>
</tr>
<tr>
<td>• Year: 1998</td>
<td></td>
</tr>
<tr>
<td>• Battery voltage: 13,6 V</td>
<td></td>
</tr>
<tr>
<td>• State of lights: OK</td>
<td></td>
</tr>
<tr>
<td>• State of light switch: OK</td>
<td></td>
</tr>
</tbody>
</table>

Solution

- **Diagnosis**: Front light fuse defect
- **Repair**: Replace front light fuse
Description of an Experience (2)

Now we have two experiences

- Each case describes one particular situation
- All cases are independent of each other

**CASE 1**

**Problem (Symptoms)**
- *Problem*: Front light doesn’t work
- *Car*: VW Golf III, 1.6 l
- *Year*: 1996
- *Battery voltage*: 13.6 V
- *State of lights*: OK
- *State of light switch*: OK

**Solution**
- Diagnosis: Front light fuse defect
- Repair: Replace front light fuse

**CASE 2**

**Problem (Symptoms)**
- *Problem*: Front light doesn’t work
- *Car*: Audi A4
- *Year*: 1997
- *Battery voltage*: 12.9 V
- *State of lights*: surface damaged
- *State of light switch*: OK

**Solution**
- Diagnosis: Bulb defect
Finding an Experience (1)

- Now: A new problem has to be solved
- We make several observations in the current situation
- Observations define a new problem
- Not all feature values have to be known

Note: The new problem is a “case” without solution part

Problem (Symptom):
- **Problem**: Break light doesn’t work
- **Car**: Audi 80
- **Year**: 1989
- **Battery voltage**: 12.6 V
- **State of light**: OK
Finding an Experience (2)

Compare the New Problem with Each Case
What is the most useful experience?

• When are two cases similar?
• How to rank the cases according to their similarity?
  \[ \Rightarrow \text{Similarity is crucial!} \]

• We can assess similarity based on the similarity of each feature
• Similarity of each feature depends on the feature value.
• BUT: Importance of different features may be different
Finding an Experience (3)

• Assignment of similarities for features values.
• Express degree of similarity by a real number between 0 and 1
• Examples:
  – Feature: *Problem*
    
    Front light doesn’t work \( \rightarrow 0.8 \) \hspace{1cm}  
    Break light doesn’t work
    Front light doesn’t work \( \rightarrow 0.4 \) \hspace{1cm}  
    Engine doesn’t start
  – Feature: *Battery voltage*  \hspace{1cm} (similarity depends on the difference)
    
    12.6 V \( \rightarrow 0.9 \) \hspace{1cm}  
    13.6 V
    12.6 V \( \rightarrow 0.1 \) \hspace{1cm}  
    6.7 V

• **Different features have different importance (weights)!**
  – High importance: Problem, Battery voltage, State of light, ...
  – Low importance: Car, Year, ...
Finding an Experience (4)

Problem (Symptom)
- Problem: Break light doesn’t work
- Car: Audi 80
- Year: 1989
- Battery voltage: 12.6 V
- State of lights: OK

Experience 1 (Symptoms)
- Problem: Front light doesn’t work
- Car: VW Golf III, 1.6 l
- Year: 1996
- Battery voltage: 13.6 V
- State of lights: OK
- State of light switch: OK

Solution
- Diagnosis: Front light fuse defect
- Repair: Replace front light fuse

Similarity computation by weighted average
\[
similarity(\text{new}, \text{case 1}) = \frac{1}{20} \times [6 \times 0.8 + 1 \times 0.4 + 1 \times 0.6 + 6 \times 0.9 + 6 \times 1.0] = 0.86
\]
Finding an Experience (5)

Problem (Symptom)
- **Problem**: Break light doesn’t work
- **Car**: Audi 80
- **Year**: 1989
- **Battery voltage**: 12.6 V
- **State of lights**: OK

Experience 2 (Symptoms)
- **Problem**: Front light doesn’t work
- **Car**: Audi A4
- **Year**: 1997
- **Battery voltage**: 12.9 V
- **State of lights**: surface damaged
- **State of light switch**: OK

Solution
- **Diagnosis**: Front light fuse defect
- **Repair**: Replace front light fuse

• Similarity computation by weighted average

\[ \text{similarity(} \text{new, case 2} \text{)} = \frac{1}{20} \times [6 \times 0.8 + 1 \times 0.8 + 1 \times 0.4 + 6 \times 0.95 + 6 \times 0] = 0.585 \]

Case 1 is more similar: due to feature “State of lights”
Problem (Symptom):
- Problem: Front light doesn’t work
- Car: Audi 80
- Year: 1989
- Battery voltage: 12.6 V
- State of light: OK

Solution:
- Diagnosis: Front light fuse defect
- Repair: Replace front light fuse

Adapt Solution:
How do differences in the problems affect the solution?

Problem (Symptoms):
- Problem: Front light doesn’t work
- ...

New Solution:
- Diagnosis: Break light fuse defect
- Repair: Replace break light fuse

Observation: case 1 is not the same as the new problem!!
An Application Scenario: Call Centre

- **Technical Diagnosis of Car Faults:**
  - symptoms are observed (e.g., engine doesn’t 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., charge battery)

- **Case-Based Diagnosis:**
  - a case describes a diagnostic situation and contains:
    - description of the symptoms
    - description of the failure and the cause
    - description of a repair strategy
  - store a collection of cases in a case base
  - find case similar to current problem and reuse repair strategy
Difference to Other Methods

- Learning in CBR does not generate an explicit generalization
- Learning = Storage of specific experiences
- Problem solving: Using specific experiences by analogy
- Contrast:
  - Generalization = Compilation of experiences
  - Learning by similarity = Interpretation of experiences
• Transformational Approach: Construction of $\alpha'$ from $\alpha$
• Derivational Approach: Construction of $\beta'$ from $\alpha$, $\beta$
Transformational Approach (1)

- Transform the solution of a similar problem
Transformational Approach (2)

- Previous experiences are stored: Problem + Solution
- General approach:
  - Search in the experience data base for experiences with similar problem descriptions
  - Find the solution belonging to the old problem description
  - (incremental) transformation of the old solution until it satisfies the demands and constraints of the new problem sufficiently well.
  - Validation of the found solution
  - If the transformation had no success or the found solution was incorrect: Search for other experiences
Derivational Approach (1)

- Transform the inference leading to the solution of a similar problem:

![Diagram]

New Problem \[\rightarrow\] partial mapping \[\rightarrow\] solved problem2

Solved problem1 \[\rightarrow\] influences \[\rightarrow\] Solution of the old problem2

Recall the inference

Solution of the new problem \[\rightarrow\] Solution of the old problem1
Derivational Approach (2)

- Previous experiences are stored:
  Problem + Solution + **Inference** (solution path)

- General approach:
  - Search in an experience data base for experiences with *similar problem descriptions*
  - Isolate commonalities between the actual problem and the experiences.
  - Determine those parts of the solution paths that can be transformed.
  - Recall the found partial solution paths in the context of the actual problem.
Definitions (Case, Case Base)

• **Given:**
  - $M$ as a underlying set
  - Problem description
  - sim: a similarity measure over $M$
  - $C$: Index set of classes

• **Definition:**
  A case $F$ is a pair $(m,c)$ with $m \in M$ and $c \in C$.

• **Definition:**
  A case base $CB$ is a finite set of cases;
  $CB = \{C_1, C_2, \ldots, C_n\}$
The Classical CBR R^4-Cycle

- This cycle shows the main activities in CBR

   - **Retrieve:**
     - Determine most similar case(s).

   - **Reuse:**
     - Solve the new problem re-using information and knowledge in the retrieved case(s).

   - **Revise:**
     - Evaluate the applicability of the proposed solution in the real-world.

   - **Retain:**
     - Update case base with new learned case for future problem solving.

---

**Problem**

- New Case

**Case Base**

- Retrieved Case

**Knowledge**

- Learned Case
- Tested/Repaired Case

**New Case**

- Solved Case

**Suggested Solution**

- Confirmed Solution

Michael M. Richter
In order to solve problems one needs knowledge. Where is it located: In knowledge containers.

A task of knowledge management is the maintenance of the containers.
Similarity Measures

- **Idea**: Numerical modeling of similarity (more or less similar)

- Along with ordinal information there is also a quantitative statement about the degree of similarity:

- **Definition**: A similarity measure on a set $M$ is a real function $\text{sim}: M^2 \rightarrow [0,1]$. The following properties can hold:
  - $\forall x \in M$: $\text{sim}(x,x) = 1$ (Reflexivity)
  - $\forall x, y \in M$: $\text{sim}(x,y) = \text{sim}(y,x)$ (Symmetry)

- Induced similarity relation:
  - $\text{sim}(x,y) > \text{sim}(x,z) \iff \text{“} x \text{ is more similar to } y \text{ than to } z \text{”}$
Nearest Neighbor

• **Definition:**
Suppose a case base $CB$, a similarity measure $sim$ and an object (problem) $p \in M$ are given, then $C \in CB$ (with $C = (m,c)$) is a nearest to $p$ if and only if:
\[ \forall (m',c') \in CB \text{ holds } sim(p,m) \geq sim(p,m') \]

• The pair $(CB,sim)$ defines by the nearest neighbor principle a classifier: to some element $p \in M$ the class of the nearest neighbor is assigned.

• **Observe:** The classifier is not uniquely defined if more than one nearest neighbor exist.
Distance Functions / Measures

• Instead of using similarity functions often one takes distance functions, the dual notion.

• **Definition:** A *distance measure* on a set $M$ is a real valued function $d: M^2 \rightarrow IR^+$.

• The following properties can hold:
  
  $\forall x \in M \quad d(x,x) = 0$ \quad (Reflexivity)

  A distance measure is *symmetrical* $\iff$

  $\forall x, y \in M \quad d(x,y) = d(y,x)$ \quad (Symmetry)

  A distance measure is a *metric* $\iff$

  $\forall x, y \in M \quad d(x,y) = 0 \rightarrow x = y$

  $\forall x, y, z \in M \quad d(x,y) + d(y,z) \geq d(x,z)$ \quad (Triangle Inequality)

• Induced similarity relation:
  
  $d(x,y) < d(x,z) \iff ”x \text{ is more similar (less distant) to } y \text{ than to } z”$
Relation between Similarity and Distance Measures

• Definition: A similarity measure $\text{sim}$ and a distance measure $d$ are called compatible if and only if
  \[ \forall x, y, u, v \in M \quad d(x,y) \geq d(u,v) \iff \text{sim}(u,v) \geq \text{sim}(x,y) \]
  (the same similarity relation is induced)

• Transformation of measures: If a bijective, strictly monotonic decreasing mapping $f: \mathbb{IR}^+ \to [0,1]$ exists with
  \[ f(0) = 1 \]
  \[ f(d(x,y)) = \text{sim}(x,y) \]
  then $\text{sim}$ and $d$ are compatible.

• Example: $f(x)=1/(1+x)$
Example 1: Hamming Distance and Simple Matching Coefficient

- Regarding similarity between two objects $x$ and $y$:
  \[ x = (x_1, \ldots, x_n) \quad x_i \in \{0, 1\} \]
  \[ y = (y_1, \ldots, y_n) \quad y_i \in \{0, 1\} \]

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

- Properties:
  - $H(x, y) \in [0, n]$  $n$ is the maximum distance
  - $H(x, y)$ is the number of distinguishing attribute values
  - $H$ is a distance measure:
    - $H(x, x) = 0$
    - $H(x, y) = H(y, x)$

- $H( (x_1, \ldots, x_n), (y_1, \ldots, y_n) ) = H( (1-x_1, \ldots, 1-x_n), (1-y_1, \ldots, 1-y_n) )$
Examples 2: Real-Valued Attributes

- Real-valued attributes $x_i, y_i \in \mathbb{R}$ for all $i$
- Generalization of the Hamming Distance to the City-Block-Metric:
  \[
  d(x, y) = \sum_{i=1}^{n} |x_i - y_i|
  \]
- Alternative metrics
  - Euclidean Distance:
    \[
    d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}
    \]
  - Weighted Euclidean Distance:
    \[
    d(x, y) = \sqrt{\sum_{i=1}^{n} \alpha_i \cdot (x_i - y_i)^2}
    \]
Weights

- The weighted Euclidean measure takes care of the fact that not all attributes are of the same importance.
- Higher weight means that the attribute has more influence on the measure and on the chance to be selected as nearest neighbor.
- Weights are a general means to reflect importance and can be applied in a wide range of measures.
- Often, the choice of good weights is a major and difficult task that can require additional learning methods.
A General Similarity Measure

• Given two problem descriptions \( C_1, C_2 \)
• \( p \) attributes \( y_1, \ldots, y_p \) used for the representation

\[
SIM(C_1, C_2) = \sum_{j=1}^{p} \omega_j \cdot sim_j(C_1, C_2)
\]

\( sim_j \) : similarity for attribute \( j \) (local measure)
\( w_j \) : describes the relevance of attribute \( j \) for the problem
Classifiers

- (CB,sim) is a classifier description
- The description is distributed over CB and sim; a change of CB or sim can lead to a different classifier
- Two extreme cases:
  a) The whole classification information is in the case base:
     - CB = \{ (p, class(p)) \mid p \in M \}
     - sim(x,y) = 1 if x=y
       sim(x,y) < 1 otherwise
  b) The whole classification information is in the similarity measure:
     - CB = “One case per class”
     - sim(x,y) = 1 if class(x) = class(y)
       sim(x,y) < 1 otherwise
No Experiences

• Often, no experiences are available. This is e.g. the case in e-commerce where no specific (problem, demand) solutions are recorded.

• Instead one records directly the similarity between problems and solutions.

• An example in e-commerce is the similarity between a demand and an available product.

• The analog of a case base is now a product base and the nearest neighbor to a demand is a product that satisfies the customer demand in an optimal way.

• In general, the products are again called cases.
Example : Selling Cars

- The problem is a customer demand:
  - Description $D$ of a demanded car
- Case base:
  - Descriptions of the available cars $C$
- The customer is offered a car $C$ where $\text{sim}(D, C)$ is maximal
- The does not refer to previous sales.
- The CBR techniques remain the same.
- However, previous sales may be used for improving the similarity measure.
How to Determine the Weights? (1)

- Basic idea: Weights reflect importance
- Importance for what? For finding a good solution.
- That means, if we consider just one attribute A, it is more important than an attribute B if the solution found by using A is better than using B.
- Better can mean different things, e.g.
  - The probability for finding the correct solution is higher
  - The expected costs for false solutions are lower
  - Etc
- These aspects should be reflected in the weights
How to Determine the Weights? (2)

- A simple situation is given for classification.
- From COWeb we take over:
- Def. : A is an attribute a some value of A, C is the class of the object O under consideration; Exp means expected value.

(i) PredictivePower (A, a, O) = Prob(C | value(A) =a)

(ii) PredPower (A, O) = Exp PredictivePower (A, a, O)| a ε dom(A))

(iii) PredPow (A) = Exp PredPower(A | O)

- This can be generalized to cost sensitive classification
- The probabilities can be estimated from the given examples.
Learning of Attribute Weights

- Goal: Improved similarity measure by weight adaptation

- There are different ways to model weights:
  - global weights: $\text{sim}(q,c) = \sum_{i=1}^{n} w_i \cdot \text{sim}_i(q_i, c_i)$
  - local (class specific) weights: $\text{sim}(q,c) = \sum_{i=1}^{n} w_{i,\text{class}(c)} \cdot \text{sim}_i(q_i, c_i)$
    - $w_{ij}$: Relevance matrix
  - case specific weights: $\text{sim}(q,c) = \sum_{i=1}^{n} w_{i,c} \cdot \text{sim}_i(q_i, c_i)$

- Adaptation of weights: Change the relevance of features for the solution.
Local Similarity Measures for Integer/Real

• Similarity often based on difference:
  – linearly scaled value ranges: \( sim_A(x,y) = f(x-y) \)
  – exponentially scaled value ranges: \( sim_A(x,y) = f(\log(x) - \log(y)) \)

• Generally, for \( f \) we claim:
  – \( f: IR \rightarrow [0..1] \) or \( Z \rightarrow [0..1] \)
  – \( f(0) = 1 \) (Reflexivity)
  – \( f(x) \): monotonously decreasing for \( x > 0 \) and monotonously increasing for \( x < 0 \)
  – Examples:
    • \( f \): symmetric
    • \( x \): query; \( y \): case
      • query is minimum demand
    • \( f \): asymmetric
    • \( x \): query; \( y \): case
      • query is maximum demand
Property: Continuous or Sudden Decrease

- Depending on the distance between two values, distinguish between decrease
  - at a **point** (→ Example a))
  - in an **interval** (→ Example b))

- Example:
  a) decrease of the whole similarity at the distance \(X\)
  b) decrease in the interval \([x_{\text{min}}, x_{\text{max}}]\)
Local Similarities for Taxonomies (1)

- Assumption: considered objects can be ordered in a Taxonomy or set of notions (tree or graph structure)

- Example:

    - The *ELSA 2000* is more similar to the *Stealth 3D* than to one of the *S3 Trio adapters* or *MGA adapters*.
Local Similarities for Taxonomies (2)

• Definition of similarity for leaf nodes:
  – assignment of a similarity value to each inner node
  – similarity values for successor nodes become larger
  – similarity between two leaf nodes is computed by the similarity value at the deepest common predecessor

• Example:

\[
sim (\text{ELSA 2000}, \text{Stealth 3D}) = 0.7
\]
\[
sim (\text{ELSA 2000}, \text{VGA V64}) = 0.5
\]
\[
sim (\text{ELSA 2000}, \text{Matrox Mill.}) = 0.2
\]
Learning of Weights with/without Feedback

• Many learning methods for both variants are known.
• Learning of Retrieval / Reuse without Feedback
  – Make use of the distribution of cases in the case base for determining the relevance of features

\[ A_1 \] is more important than \( A_2 \)

• Learning with Feedback
  – Correct or false case selection / classification result leads to correction of weights
Learning of Weights without Feedback

- Determination of class specific weights:
  - Binary coding of the attributes by
    - Discretization of integer and real valued attributes
    - Transformation of symbolic attributes to $n$ binary attributes
  - Suppose
    - $w_{ik}$ is the weight for attribute $i$ for class $k$
    - $\text{class}(c)$ is the class (solution) in case $c$
    - $c_i$ the attribute $i$ of case $c$
  - Set: $w_{ik} = \text{Prob}(\text{class}(c)=k | c_i)$
    Conditional probability for a class $k$ of $c$ under the condition of attribute $c_i$.
  - Estimate of the probabilities is done using “samples” of the case base.
Learning of Weights with Feedback

- Correct or false classification leads to a correction of the weights:
  \[ w_{ik} := w_{ik} + \Delta w_{ik} \]
- Different variants for determining the weight changes:
  - Approach of Salzberg (1991) for binary attributes:
    - Feedback = positive (correct classification):
      - Weight of coincident attributes is increased
      - Weight of differing attributes is decreased
    - Feedback = negative (false classification):
      - Weight of coincident attributes is decreased
      - Weight of differing attributes is increased
  - Here: \( \Delta w_{ik} \) is constant.
Feedback Method: Gradient Descent

- We consider the classification error
- Experimentally to determine: e.g. leave-one-out test
- Goal: Optimizing the values of the weights s.t., e.g. classification quality becomes optimal.
- Change of weights in direction of steepest descent.
- Compare: Backpropagation
Algorithm for Gradient Descent

1. Initialize the weight vector $\mathbf{w}$ and die learning rate $\lambda$

2. Compute the error $E(\mathbf{w})$ for the weight vector $\mathbf{w}$

3. While not stop criterion Do

   a) Learning step:  
   \[ \forall a \hat{w}_a := w_a - \frac{\partial E}{\partial w_a} \cdot \lambda \]

   b) Compute:  
   \[ E(\hat{\mathbf{w}}) \]

   c) If $E(\hat{\mathbf{w}}) < E(\mathbf{w})$ Then $\mathbf{w} := \hat{\mathbf{w}}$ Else $\lambda = \frac{\lambda}{2}$
Learning Rate

- Choice of the learning rate $\lambda$
Further Parameters of the Algorithm

• Choice of the stop-criterion:
  – Fixed number of steps
  – Minimal change of the error function
  – Minimal change of the weights
  – Minimal change of the learning rate
Retrieval Methods

- There are many retrieval methods
- Trivial but too slow: Sequential Retrieval
- Important: k-dimensional binary search tree (Bentley, 1975).
- Idea: Decompose data (i.e. case base) iteratively in smaller parts
- Use a tree structure
- Retrieval:
  - Searching in the tree top-down with backtracking
Example: kd-Tree

CB={A,B,C,D,E,F,G,H,I}
Definition: kd-Tree

• Given:
  – k ordered domains \( T_1, ..., T_k \) of the attributes \( A_1, ..., A_k \),
  – a base \( CB \subseteq T_1 \times ... \times T_k \) and
  – some parameter \( b \) (bucket size).

A \textit{kd-tree} \( T(CB) \) for the base \( CB \) is a binary tree recursively defined as:

– if \( |CB| \leq b \): \( T(CB) \) is a leave node (called \textit{bucket}) which is labelled with \( CB \)

– if \( |CB| > b \): \( T(CB) \) then \( T(CB) \) is a tree with the properties:
  • the root is labelled with an attribute \( A_i \) and a value \( v_i \in T_i \)
  • the root has two kd-trees \( T_{ \leq } (CB_{ \leq } ) \) and \( T_{ > } (CB_{ > } ) \) as successors
  • where \( CB_{ \leq } := \{ (x_1, ..., x_k) \in CB \mid x_i \leq v_i \} \) and \( CB_{ > } := \{ (x_1, ..., x_k) \in CB \mid x_i > v_i \} \)
Properties of a kd-Tree

• Ein kd-tree partitions a base:
  – the root represents the whole base
  – a leave node (bucket) represents a subset of the base which is not further partitioned
  – at all inner nodes the base is further partitioned s.t. base is divided according to the value of some attribute.
Retrieval with kd-Trees

Idea for an algorithm:

1. Search the tree top-down to a leave
2. compute similarity to objects found
3. Determine termination by BWB-test
4. Determine additional candidates using a BOB-test
5. If overlapping buckets exist then search in alternative branches (back to step 2)
6. Stop if no overlapping buckets

BWB test
Ball-within-Boands

BOB test:
Ball-overlap-Boands
BOB-Test: Ball-Overlap-Bounds

Are there more similar objects than the $m$-most similar object found in the neighbor subtree?

- $n$-dimensional hyper ball
- Boundaries of the actual node
- Overlap
- $m$-most similar object in scq
BWB-Test: Ball-Within-Bounds

Is there no object in the neighboring subtree that is more similar than the m-most-object?

n-dimensional hyper ball

boundaries of the actual node

m-most-similar object in scq
Case Based Learning

• Given: A sequence of training cases $C_1, C_2, ..., C_k$

• Definition:

• *Case based learning* determines, starting with an empty case base $CB_0$ and an initial measure $sim_0$ a sequence of tuples: $(CB_1, sim_1), (CB_2, sim_2), ..., (CB_k, sim_k)$ with $CB_i \subseteq \{C_1, ..., C_i\}$

   With each occurrence of a training case can happen:
   
   – the case base is changed (e.g. by including the case)
   
   – the measure is changed
   
   – both are changed
Information in Similarity Measures

• Goal: Measure the knowledge contained in similarity measures:

• Def: The measure \( sim_1 \) of a case based system \( S_1 = (CB, sim_1) \) is relative to \( CB \) better informed than the measure \( sim_2 \) of the system \( S_2 = (CB, sim_2) \), if \( S_1 \) classifies more cases correctly than \( S_2 \).

• Def: A case base CB of a case based system \( (CB, sim) \) is called minimal if there is no case base \( CB' \subset CB \) s.t. \( (CB', sim) \) classifies at least so many cases correctly than \( (CB, sim) \) does.
Instance-based Learning

• Some form of case based learning that adapts only the case base i.e. the similarity measure remains constant.

• Case representation = Attribute-value pairs.

• For real attributes often:

• Geometric interpretation:
  – Distance measure = Euclidean distance:  \[ d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \]
Three Incremental Learning Algorithms IB1, IB2, IB3

• Given:
  – Similarity measure: Euclidean distance
  – Sequence of training examples $C_1, C_2, ..., C_n$

• Wanted:
  – case base $CB = CB_n$

• Approach:
  – Determine a sequence of case bases $CB_1, ..., CB_n$

• Three Variants:
  – IB1: Takes all cases into the case base
  – IB2: Takes cases only if the actual case base performs a misclassification
  – IB3: Takes cases only if the actual case base performs a misclassification and removes in addition „bad“ cases.
IB1

Let $T = (F_1, \ldots, F_n)$ be the given sequence of training examples

algorithm:

\[ CB = \{ \} \]

FOR $i=1$ to $n$ DO

\[ CB := CB \cup \{ F_i \} \]

Problem:

- case base grows fast.
- If IB1 would be used in the Retain-Phase of the CBR-Cycle then the case base would grow each time a problem is solved
Let $T = (F_1, ..., F_n)$ be the given sequence of training examples

algorithm:
CB={}
FOR i=1 to n DO
    Set $C_i = (p,c)$ (* p = problem; c = class*)
    Choose from CB a case $C’=(p’,c’)$ which is a nearest neighbor to $C_i$.
    IF $c \neq c’$ THEN (* Failure *)
        CB := CB \cup \{F_i\}
Properties of IB2

- IB2 depends on presentations sequence!
  - Although a training case is correctly classified in the training phase and is therefore not taken into the case base it may very well happen that this case is misclassified in the final case base.

- IB2 stores much less cases.
- In experiments: IB2 has a classification performance almost as good as IB1.
Problem for IB2

• Dealing with noisy data is problematic
  – Noisy data lead often to classification errors
  – If noisy data enter the case base:
  – Lower classification quality!

• Goal: IB3 algorithm should be able to handle noisy data.

• Extension in IB3: For each case a statistics for the classification quality is made.

\[
CQ(F) = \frac{\text{Number of correct classifications by } F}{\text{Total number of problems classified by } F}
\]
IB3 (1)

Let $T = (F_1, \ldots, F_n)$ be the given sequence of training examples.

**Algorithm:**

\begin{itemize}
  \item $CB = \{\}$
  \item \textbf{FOR} $i=1$ to $n$ \textbf{DO}
    \begin{itemize}
      \item Let $F_i = (p, c)$ (* $p =$ problem; $c =$ class*)
      \item $CB_{acc} := \{ F \in CB \mid \text{acceptable}(F) \}$
      \item IF $CB_{acc} \neq \{\}$
        \begin{itemize}
          \item THEN
            \begin{itemize}
              \item choose from $CB_{acc}$ a case $F' = (p', c')$ which is a nearest neighbor to $F_i$.
            \end{itemize}
        \end{itemize}
      \item ELSE
        \begin{itemize}
          \item choose randomly between 1 and $|CB|$
          \item choose from $CB$ a case $F' = (p', c')$ which has the $j$-largest similarity to $F_i$.
        \end{itemize}
      \item IF $c \neq c'$ THEN $CB := CB \cup \{F_i\}$
    \end{itemize}
\end{itemize}
IB3(2)

Continuation of the algorithm (still in the outer loop):

FOR ALL \( F^* = (p^*, c^*) \) from \( CB \) with \( \text{sim}(p, p^*) \geq \text{sim}(p, p') \) DO

update the statistics \( CQ(F) \) depending on \( c \) and \( c^* \):
- for correct classification \((c = c^*)\) increase denominator and numerator of \( CQ(F) \);
- for false classification \((c \neq c^*)\) increase denominator of \( KG(F) \).

IF \( F^* \) is significantly bad THEN

\[ CB := CB \setminus \{F^*\} \]
Criteria: Acceptable and Significantly Bad

- Different statistical measures for the classification quality are possible.
- A simple variant:
  
  \[
  \text{acceptable}(F) \text{ iff } CQ(F) > \theta_1
  \]
  
  \[
  F \text{ is significantly bad iff } CQ(F) < \theta_2
  \]
  
  \(\theta_1\) and \(\theta_2\) are parameters with which the algorithm can be controlled.
## Databases vs. CBR

<table>
<thead>
<tr>
<th>Database System</th>
<th>CBR System</th>
</tr>
</thead>
<tbody>
<tr>
<td>• simple search ⇒ “all or nothing”</td>
<td>• using same database but search for most similar cases</td>
</tr>
<tr>
<td>• often too many hits (underspecification) or no hits at all (overspecification)</td>
<td>• system can be told to show only, e.g., 10 cases by descending order</td>
</tr>
<tr>
<td>• no specific domain knowledge used for the search</td>
<td>• considers domain knowledge for search by using similarity measures, e.g., spatial or geographical relations</td>
</tr>
<tr>
<td>• pure database applications cannot be used for online consulting</td>
<td>• online consulting is the power of a CBR system</td>
</tr>
</tbody>
</table>
Example: SIMATIC Knowledge Manager

- Customers use a www site to describe the problem in terms of
  - a textual query
  - some information of the domain and the devices involved.

- By similarity based retrieval the most useful document is presented to the customer.

- There are three types of documents:
  - FAQ’s: Contain well established knowledge
  - User information notes: Are less reliably
  - Informal notes: Draft notes which give informal hints and may be unreliable.
Title: Order numbers of CPUs with which communication is possible.

Question: Which order numbers must the S7-CPPs have to be able to run basic communications with SFCs?

Answer: In order to participate in communications via SFCs without a configured connection table, the module needs the correct order number. The following table illustrates which order number your CPU must have to be able to participate in these S7 homogeneous communications.
SIMATIC Knowledge Manager

Case Server

<table>
<thead>
<tr>
<th>Similarity model</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Structure</td>
<td>Order No.</td>
<td>Dictionary</td>
</tr>
<tr>
<td>Information</td>
<td>Relation</td>
<td>Information</td>
</tr>
<tr>
<td>about the</td>
<td>order numbers</td>
<td>Entities</td>
</tr>
<tr>
<td>Structure of the</td>
<td>- product names</td>
<td>Similarities</td>
</tr>
<tr>
<td>SIMATIC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Information</td>
<td></td>
<td></td>
</tr>
<tr>
<td>System</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Documents in the</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Information</td>
<td></td>
<td></td>
</tr>
<tr>
<td>System</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

www.ad.siemens.de
Positioning the CBR Method

- Knowledge, Models
- Closed Knowledge Model
- Expert Systems
- Increasing Knowledge Centralization

- Case Base
- Knowledge Model
- Verified Solution
- New Case
- Similar Case
- Redraw
- Adapted Solution
- Retained Solution
- Retained
- Retrieve
- Reuse
- Revise

- Database
- Open Database
- CBR Systems
- Increasing Example Orientation
- Database Systems
Tools

- Purpose of the tools:
  - Possibilities of
    - Defining attributes and their domains
    - Defining similarity measures (local similarities, weights)
    - Entering case
    - Nearest neighbor retrieval
- The tool should return the nearest neighbor(s)
- There are several such (quite efficient) tools available
- We will shortly discuss the tool mycbr
- See (This contains a tutorial and the tool can be downloaded):
  - http://mycbr-project.net/
MyCBR

• It uses attribute value descriptions
• Similarity Modeling in mycbr has global similarity measure functions composed of local measures of each slot (type of the attribute).
• One can choose between the following similarity modes (depending on the type of you slot):
  • Integer, Symbol: Taxonomy, ordered, symbol, table,…String: For each type one can define subtypes.
  • The *advanced* similarity mode can be chosen for the slot types integer and float. One should use this in case the similarity measure function cannot be represented by the *standard* similarity mode.
Summary

- General scheme
- Transformational and derivational analogy
- Similarity measures, distances
- Retrieval
- Case based learning
- IB1, IB2, IB3
- Learning of weights with/without feedbacks
Recommended References