Case-Based Reasoning

In this lecture, we turn to another popular form of reasoning system: case based reasoning (CBR)

Unlike Rule-based systems and Fuzzy Logic, CBR does not use any rules or logical deduction (inference).

Instead, CBR relies on the process of reasoning by ***analogy***:

Here is a new problem. How can we solve it?

Compare the new problem to all the problems we solved in the past. Which one does it resemble the most?

Reuse the solution of the most similar past problem by adapting it to get a solution for the new problem.

And finally, file away the new problem and its solution for reference in solving future problems.

Case-Based Systems

Knowledge is stored not as rules but as *records of cases*.
Basic approach:

- Identify certain important aspects ("attributes") of "cases". These might be particular circumstances which make up a situation, or particular features of objects.
- Store the details (values of attributes) of known cases in separate records (perhaps using a relational database). The values stored may be numerical "codes" (indices). The known cases may be from actual experience or may be invented.
- Given partial information about a subsequent case, we can look up the case library to match what we know with one or more stored cases, the details of which should provide more information pertinent to the new case.

Case-Based Systems (2)

So a case-based system will involve:

1. A library of cases.
2. A means of using key elements of the present situation to find and retrieve the *most similar* case(s) from the library. Note that this may incorporate a sophisticated way of judging similarity.
3. A means of drawing conclusions. Because we may not find an exact fit, some judgement may have to be exercised. This stage is known as "adaptation".

Case-Based System Examples

1. **Bank Loans**. Banks must make judgements about appropriate loans. Over time, experience is gained of good and bad loans. A case library may contain records:

- details of the customer, e.g.:
  - age,
  - income,
  - postcode,
  - marital status,
  - employment history
  - amount of loan granted,
  - amount of regular repayments
- details of the outcome, e.g.:
  - customer defaulted,
  - customer was late with repayments,
  - no problems

When a customer asks for a loan, the bank could look up similar customers and loans in the case library, and arrive at a judgement about the new loan.

Case-Based System Examples (2)

2. **Automatic e-mail interpretation and routing.**

The task is to receive, interpret, and find an appropriate way to deal with incoming electronic messages from customers.

First set up a case library: invent example messages with certain identified attributes.

When a message comes in, computer identify its attributes, look up the case library for matching ones, and discover the appropriate action, which may be:

- Respond to the message automatically with an appropriate set response, or
- Forward the message to the appropriate staff member.

(Reference: US Patent 6182059)

Case-Based System Examples (3)

3. **Property Valuation.**

Significant attributes used (there could be others):

- living area
- number of bedrooms
- number of bathrooms
- architectural style
- construction method
- age
- location
- date of valuation
- type of heating
- type of garage
- area of grounds

The case library would consist of many records for individual properties, each record containing also an estimated value.

Case-Based System Example (4)

Given the details of a house, the system may retrieve details of similar houses. In this example, exact matches will not occur, so this would require an algorithm to measure similarity. Moreover, the different characteristics might not carry equal weight in making these judgements.

Once the best matching properties have been found, there may be a further process --- some compensation may need to be made for differences between the property being valued and the retrieved cases in order to estimate the value. This is the "adaptation" stage.

In general, adaptation might not be necessary. But if so, it would also be an "expert" task. But compensation for too many differences will reduce confidence in the conclusion.

Structure of a Case Based System

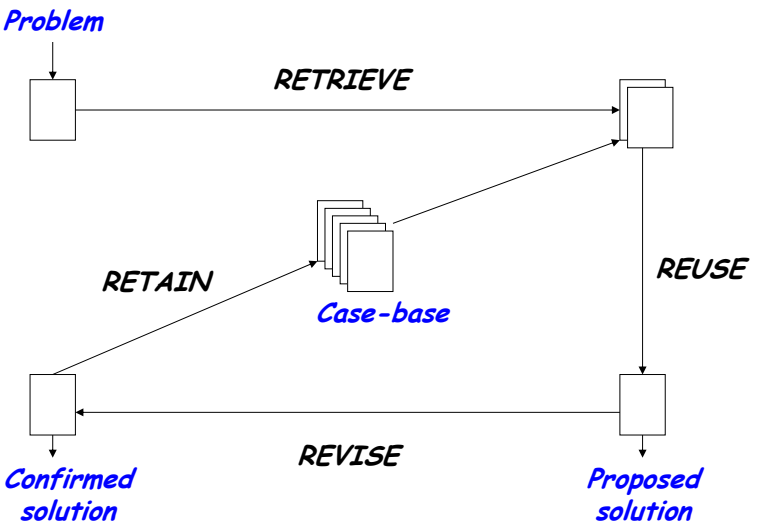
"A case-based reasoner solves new problems by adapting solutions that were used to solve old problems." [Riesbeck & Schank, 89]

A case-based system relies upon a library of cases representing old problems and their solutions.

To solve a new problem, the system does the following:

- RETRIEVE the most similar case(s) from the library
- REUSE the case(s) attempt to solve the problem
- REVISE the proposed solution if necessary (*adaptation*)
- RETAIN the new solution as part of a new case.

Structure of a Case Based System



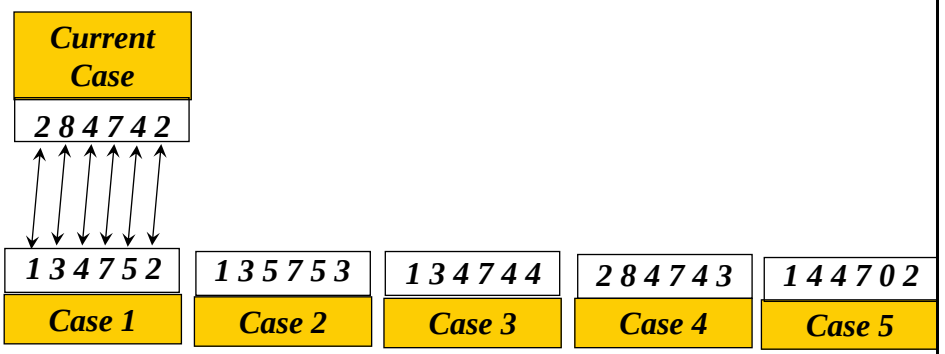
Retrieval and Adaptation

The *retrieval* step requires a means of using key elements of the present situation to find and retrieve the *most similar* case(s) from the library. Note that this may incorporate a (more or less) sophisticated way of judging similarity.

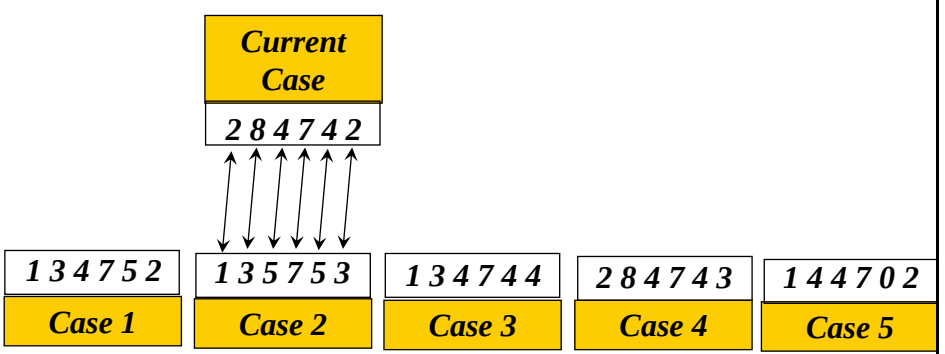
The system might use a variant of the "nearest neighbour" algorithm – see later.

Because we may not find an exact match, we may need to exercise some judgement to revise the solution that was found so as to fit it to the present problem. This process is known as "adaptation".

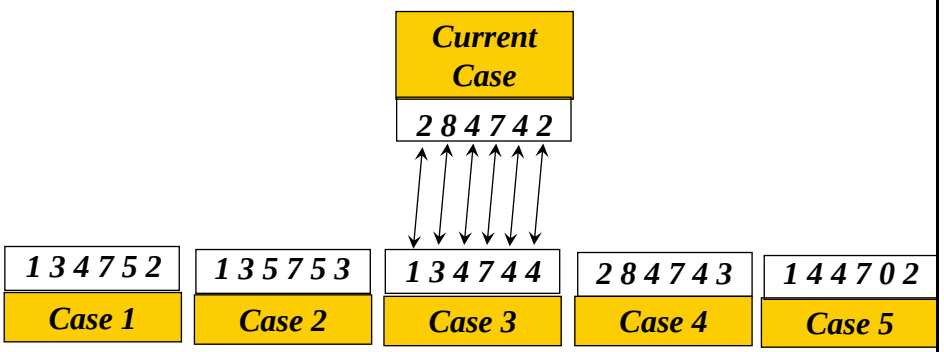
The Retrieval Process



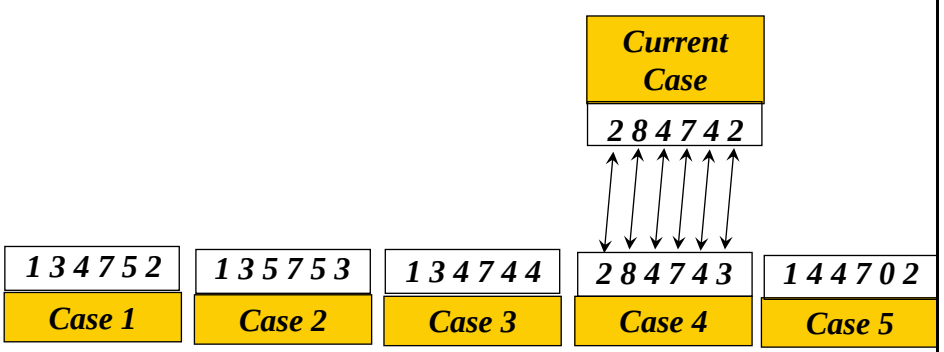
The Retrieval Process (2)



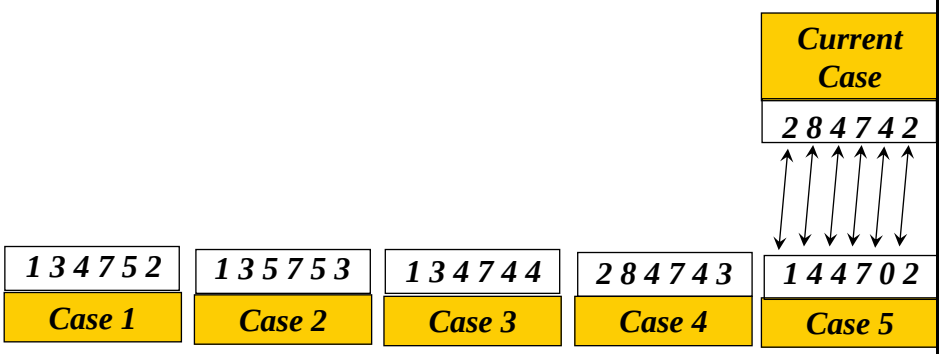
The Retrieval Process (3)



The Retrieval Process (4)

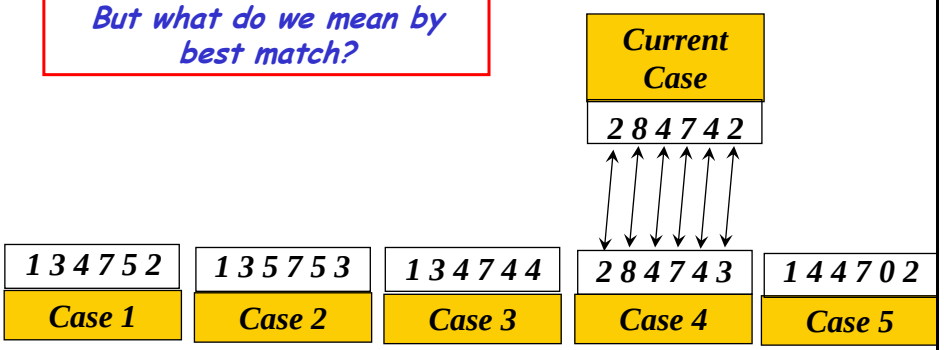


The Retrieval Process (5)



Bingo! Best Match

But what do we mean by best match?



"Best Match": Measuring Similarity

One way to measure similarity of cases is to use the "nearest neighbour" algorithm.

To illustrate:

Suppose that we have a sick soya plant, and we wish to discover which of a number of known specimens of sick soya plants it is most like (so that we can perhaps identify the cause of the sickness, and fix it).

Soya plant symptoms

Figure 13. The foliar symptoms of brown stem rot of a diseased plant.

Figure 9. Manganese deficiency symptoms appear as yellowing between the veins.

Figure 17. Leaves showing symptoms of bacterial blight.

"Nearest Neighbour" Algorithm

Choose (let's say) three characteristics of the leaves that can be represented as numbers:

1. Amount of the leaf that is covered by the discolouration
2. Lightness of the discoloured parts of the leaf
3. Lightness of the remaining parts of the leaf.

Figure 13. The foliar symptoms of brown stem rot of a diseased plant.

ure 9. Manganese deficiency symptoms appear as yellow-ing between the veins.

eaves showing symptoms of bacterial blight.

"Nearest Neighbour" Algorithm (2)

Suppose that our sick plant has the index values:

specimen: coverage - 8
lightness1 - 4
lightness2 - 6

"Nearest Neighbour" Algorithm (3)

And suppose that the first two cases in our case memory to be matched are:

case 1: coverage - 10
lightness1 - 7
lightness2 - 6

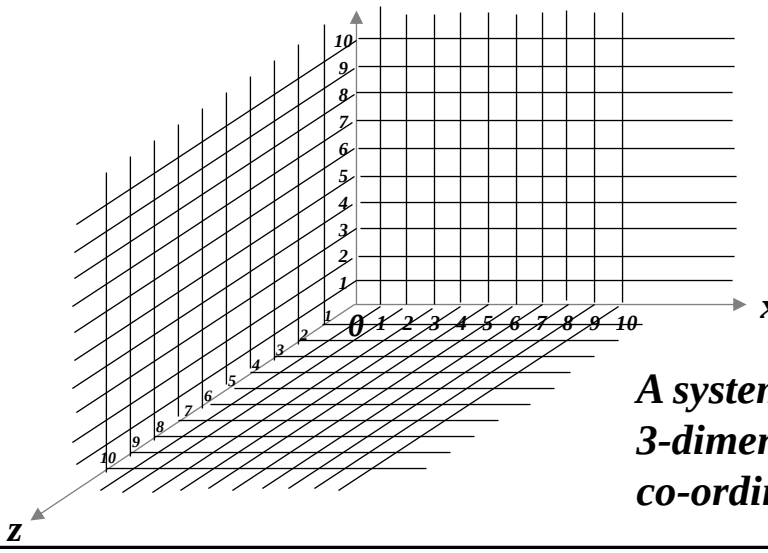
case 2: coverage - 2
lightness1 - 3
lightness2 - 9

"Nearest Neighbour" Algorithm (4)

These can be treated as coordinates of points in three-dimensional space:

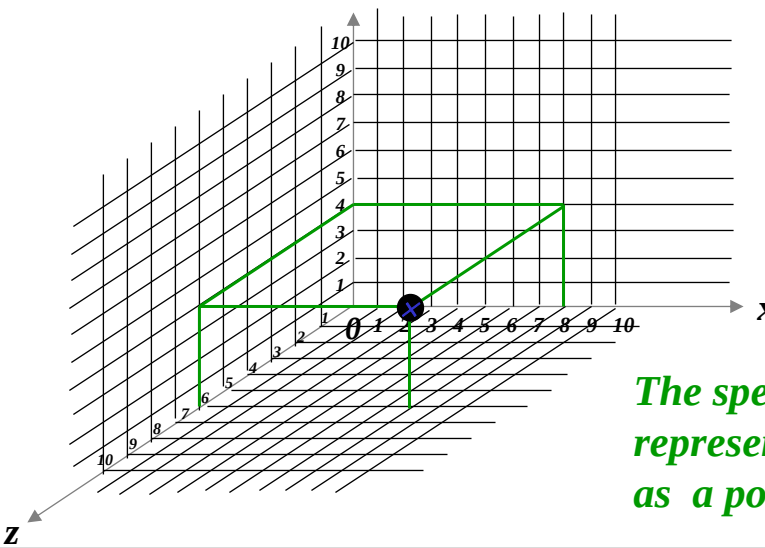
x, y, z coordinates of specimen: (8, 4, 6)
x, y, z coordinates of case 1: (10, 7, 6)
x, y, z coordinates of case 2: (2, 3, 9)

"Nearest Neighbour" Algorithm (5)



A system of 3-dimensional co-ordinates

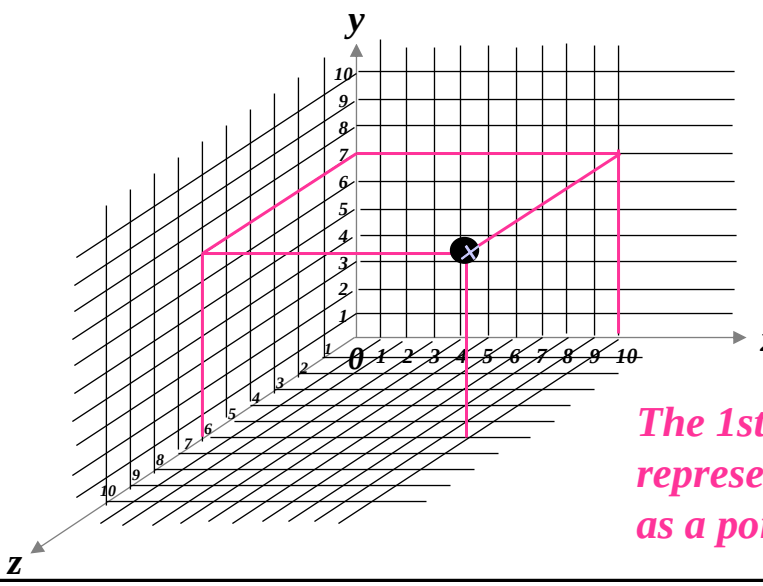
"Nearest Neighbour" Algorithm (6)



specimen

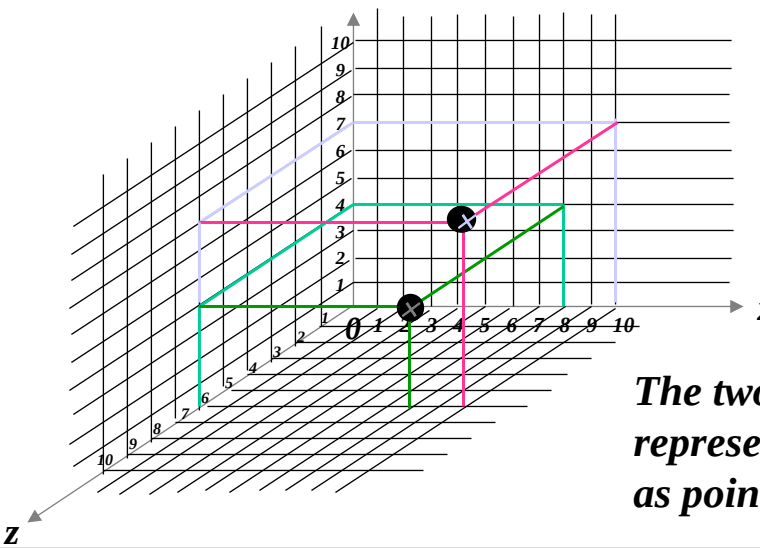
The specimen represented as a point

"Nearest Neighbour" Algorithm (7)



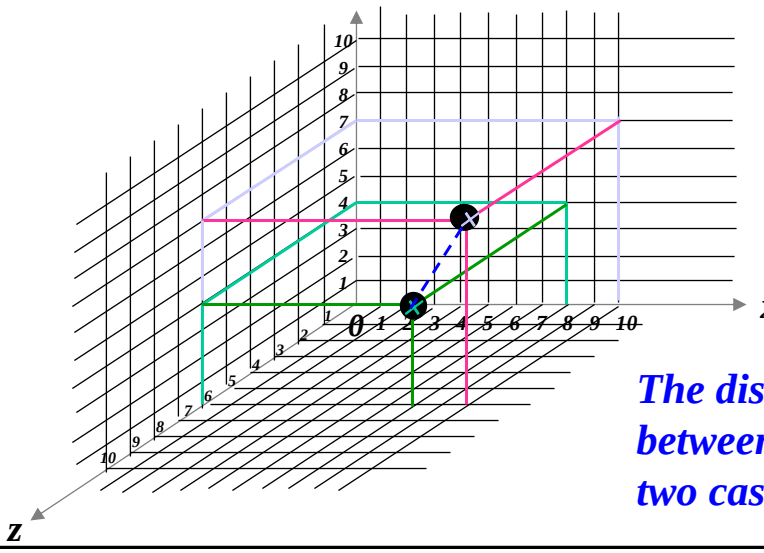
The 1st case represented as a point

"Nearest Neighbour" Algorithm (8)

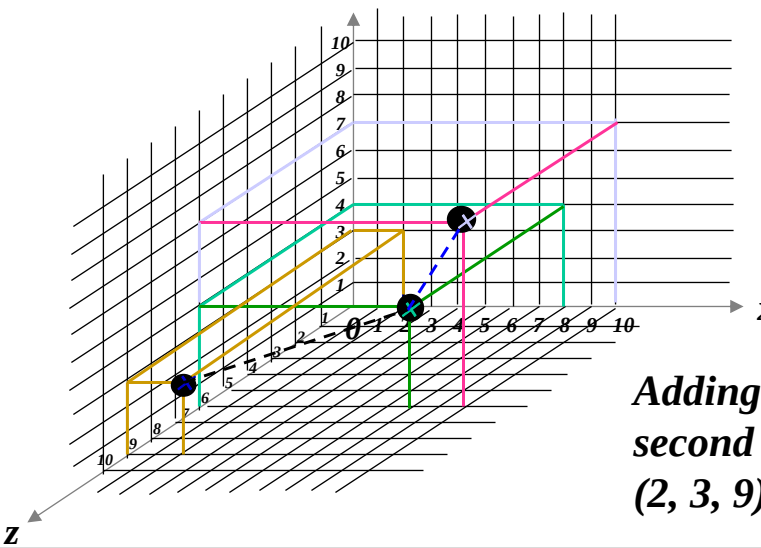


The two cases represented as points

"Nearest Neighbour" Algorithm (9)



"Nearest Neighbour" Algorithm (10)



"Nearest Neighbour" Algorithm (11)

There is a simple formula that tells you the distance between two points in 3-dimensional space.

The formula is the Euclidean distance:

$$\sqrt{[\ (x1 - x2)^2 + (y1-y2)^2 + (z1-z2)^2\ ]}$$

To find out whether the specimen case is more similar to case 1 or to case 2, you simply calculate the two distances, and pick the smaller of the two.

"Nearest Neighbour" Algorithm (12)

To find out which of a whole series of cases the specimen case is most similar to, calculate the distance from the specimen case to each of them, and pick the smallest figure.

Example Problem: Buying a Car

Attributes: (*Numbers in brackets are the indices – numerical codes used to measure similarity)*

Price

1000(1) 2000(2) 3000(3) 4000(4) 5000(5)

Year

1998(1) 1999(2) 2000(3)

Doors

3-door(1) 5-door(2)

Colour

Metallic Green(1) Purple(2) Black(3) Red(4)

Example

Question: What make of car should I buy?

Requirements (current case):

Colour: Metallic Green (1),

Year: 1998 (1),

Doors: 3-door (1),

Price: <=£1000 (1)

Example

Case Library:

Colour	Year	Doors	Price	Car
4	3	2	1	Nissan Micra
2	2	1	3	VW Polo
1	2	2	3	VW Golf
1	2	1	4	Honda Civic

Which matches the closest?

Use Euclidean Distance formula:

$$\sqrt{[\ (x1 - x2)^2 + (y1-y2)^2 + (z1-z2)^2 + \ldots\ ]}$$

Example

Difference between current case and case1:

- $\sqrt{[\ (1-4)^2 + (1-3)^2 + (1-2)^2 + (1-1)^2\ ]} = \sqrt{(9+4+1+0)} = \sqrt{14}$

Difference between current case and case2:

- $\sqrt{[\ (1-2)^2 + (1-2)^2 + (1-1)^2 + (1-3)^2\ ]} = \sqrt{(1+1+0+4)} = \sqrt{6}$

Difference between current case and case3:

- $\sqrt{[\ (1-1)^2 + (1-2)^2 + (1-2)^2 + (1-3)^2\ ]} = \sqrt{(0+1+1+4)} = \sqrt{6}$

Difference between current case and case4:

- $\sqrt{[\ (1-1)^2 + (1-2)^2 + (1-1)^2 + (1-4)^2\ ]} = \sqrt{(0+1+0+9)} = \sqrt{10}$

BEST MATCH IS CASE 2 or 3 (as there is the smallest difference between these cases, and what we require) – recommend Polo or the Golf

Issues in Measuring Similarity

In this example we took a very naive approach to comparing attributes and measuring similarity:

All attribute values were represented as numerical codes (indices).

Similarity between two attribute values is measured as the numerical difference between the corresponding indices.

This seems appropriate for attributes that are essentially numerical in nature (price, year)

But is this an appropriate way to compare non-numerical attributes (such as colour or doors)?

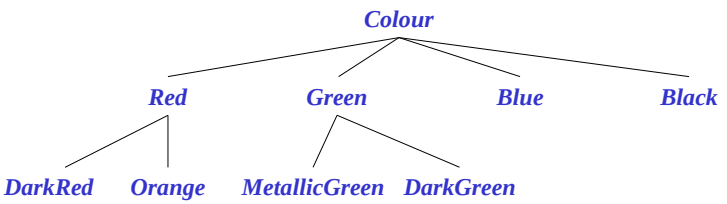
In practice, CBR tools provide various methods to define customized similarity measures for different attributes.

For example, the tool myCBR (a plugin for the Protege *ontology editor*) provides a selection of similarity modes for defining how to compare attribute values. These include numerical modes (integer, float) and symbolic modes (table, taxonomy, ordered, etc). Let's take a look at one of these: taxonomy...

Using Taxonomy to Compare Colours

Taxonomy-based comparison is suitable for comparing whose values are non-numerical but can be classified into groups whose members resemble one another. This classification may be hierarchical. Colour is an example of such an attribute.

A taxonomy of colours must first be defined, for example:



This taxonomy is used to compare the similarity of colours. For example, DarkRed is very similar to Orange, as they share the same parent. They are both also very similar to Red (their parent). They are both very different from MetallicGreen.

Adaptation

Once we have found the best match, there may still be work to do – to "adapt" the case that has been found if necessary, and to draw some conclusion about our originally given case.

- This may involve eliciting more information about the given case.
- It may involve some judgement or expertise.

Systems exist where this adaptation stage is carried out by a completely independent expert system.

Alternatively, adaptation may need to be carried out by a human expert.

Issues in Adaptation

Adaptation is often seen as the most difficult step in implementing a CBR system. The level of difficulty depends on the kind of system being implemented.

Systems using CBR for classification or identification may not require any adaptation step. In such systems, either the retrieved solution is an exact match, or no solutions are retrieved. Example: soya plant disease identification system.

However, where CBR is used for design or planning activities, it is much more likely that retrieved solutions will only be approximate matches. Adaptation will therefore be required. We will look at one such system...

Adaptation using an ANN

(Lotfy and Mohammed, 2002)

A CBR system was created to estimate the cost of constructing steel buildings. The case library consisted of records of other buildings that had been constructed in the past, together with their cost.

Problem: it is very unlikely that two buildings are exactly identical. Therefore, in using the system to estimate the cost of a proposed new building, only approximate matches can be retrieved. Adaptation is therefore required to solve the new case.

Solution: an artificial neural network is used for adaptation. The retrieved solutions are used to train the ANN. The trained ANN is then used to solve the new case.

Updating the Case-Base

Note that there is a problem about updating the case-base with adapted cases.

Since the new case isn't exactly like any of the cases in the case-base, it can't really be said to have been solved by the expert judgement that was used to build the case-base in the first place.

> There is a real chance that the conclusion that the system came to is wrong in this case.

If wrongly concluded cases are added to the case-base, it becomes progressively degraded.

Typically, the procedure is to put fresh cases into a special file, and have a Domain Expert pass judgement on them before they are added to the case-base.

Practical Aspects: User Interface

In an interactive CBR system, what kind of interface would be appropriate?

The system needs to obtain values for the significant attributes.

Maybe just a simple form. But …

the user may require guidance on what is significant/relevant

some fields may be optional

the relevance of some attributes may depend on values of other attributes.

So the interface may be structured, to prompt for just those values that are relevant.

Practical Aspects: Indexing

Looking up our case library may be a substantial task:

- We may have to inspect every stored case in order to find the best match
- Where the case library is large, some *indexing* mechanism may be necessary (so that searches can be focused towards cases which are likely to fit, and away from cases which are unlikely to fit).
- Methods for setting up an indexing mechanism are well-developed in the database world. Artificial neural networks may also be used to provide an indexing procedure.

Use of Case-Based Systems

It has been argued that the case-based approach is easier for people to work with (than the rule-based approach), and that building C-B systems is less time-consuming. Reasons:

- "What do we do now?" Well, what did we do before?
- Less formal analysis is required. Causal relationships do not have to be identified.
- Less formal logic is required.
- Expansion (indeed knowledge acquisition generally) is easier --- just add more cases.
- No need for 'general' rules (less abstraction).
- Uses reasoning by analogy (as people often do).

Use of Case-Based Systems (2)

But in some respects the work may be harder:

- We must identify the significant attributes whose values are to be used in the lookup procedure.
- We must be able to find values for these attributes, to be used as keys when looking up.
- When given a new case we must be able to identify these attribute values so that we can look up the case library.
- We must decide on an algorithm to measure similarity.
- We must decide whether adaptation will be necessary, and if so, how it is to be done.
- And there may be practical difficulties with the looking up …

Use of Case-Based Systems (3)

Case-based reasoning systems are claimed to be one of the most successful applications of Artificial Intelligence technology. Some applications include:

- Helpdesks
- Diagnostic systems
- Identifying fraud

Tools for case based reasoning come in the form of a "shell", which can be customized to set up a CBR system.

Examples: CBR3, CaseAdvisor, myCBR, jColibri

When to Use Case-Based Systems

Indicators that a problem domain is suitable for a CBR approach:

1. There are existing records of previously solved cases.
2. Historical cases are often referred to in solving new cases (as happens in law, for example).
3. Human experts tend to talk about the problem domain in terms of examples rather than general rules or policies.
4. The problem domain is not well-defined or well-understood. (e.g. Machine translation, where Google's statistical approach appears to be succeeding where older, analytical approaches failed)
5. Experience is seen as being at least as valuable as textbook knowledge.