Knowledge representation (INFO0049-1)

Exercise session 7

31 Mar 2015

Try to draw search trees wherever possible to see how prolog executes a query

---

1. Sudoku puzzle

1. A sudoku puzzle is a 9x9 table. In the final solution for the puzzle, each of the 81 cells in the table should contain an integer number from 1 to 9 such that:
    a. no row contains the same number twice (or more)
    b. no column contains the same number twice
    c. no block contains the same number twice (a *block* is one of the nine 3x3 subtables of the big table; see the thick lines in the example below).

    In the initial puzzle, some of the numbers in the cells are given. To solve the puzzle, you have to find all the remaining numbers. Write a Prolog program that solves such a puzzle.

    1. Think about a high-level strategy to solve the problem. For now, don't worry about efficiency too much (that comes later).
    2. Think about a good representation of (the solution of) a sudoku puzzle.
    3. Write a Prolog program that solves a given puzzle, i.e. a program that fills in the missing numbers in the Sudoku. Do this by transforming your high-level strategy into Prolog code.
    4. Write prolog predicates to solve the lower level tasks.

---

2. Classification problem

2. Suppose you have some bottles of wine from three different wine farmers in total. For most bottles you know from which wine farmer they are (these bottles form the so-called *train data*). However, for some of the bottles you do not know from which farmer they are (this is the *test data*). Now you want to develop a system that classifies the bottles of wine from the test data: the system should specify whether the given bottle is from farmer 1, 2 or 3. As input, you are given for each bottle the result of a 'chemical analysis' of the wine in the bottle, this is a list with 13 numbers representing 13 various properties of the wine in that bottle.

   We will solve this problem with a technique called *Nearest Neighbors*. If we want to classify a bottle b1 from an unknown farmer (a bottle from the test data), we can look for the bottle b2 in the train data that is most similar to bottle b1. We will then assume that b1 is from the same farmer as b2. This technique is called 1-Nearest Neighbor. We could also look for the 5 bottles in the train data that are most similar to bottle b1, check from which farmer most of these 5 bottles are and then assume that bottle b1 is also from that farmer. This is called 5-Nearest Neighbors.

   To be able to use this technique, we need to have a measure of distance between two bottles of wine b1 and b2. If we have such a measure, we then define the bottle that is 'most similar' to a given bottle b1 as being the bottle for which the distance to b1 is the smallest. For the distance between two bottles we will actually use the distance between the two lists associated with the bottles (as said before, for each bottle we have a list with 13 numbers representing properties of the wine in the bottle).

   WineData.pl contains 125 traindata/2 facts. Each of these facts corresponds to 1 bottle for which we know the farmer. The first argument of such a fact is always a chemical analysis (a list of 13 numbers), the second argument is the corresponding farmer (1, 2 or 3).
   WineData.pl also contains 39 testdata/2 facts. Each of these facts corresponds to 1 bottle for which we do not know the farmer. The first argument of such a fact is always a bottle-number (b1 to b39), the second argument is a chemical analysis.

   - Write a predicate that calculates a distance measure between two chemical analyses (e.g. Manhattan distance, Euclidean distance, ...). Note that this is a distance measure in the 13-

dimensional space (since a chemical analysis is a list with 13 elements).

- Write a predicate that, given a bottle in the test data, classifies this bottle (i.e. determines from which farmer it is). Use 5 Nearest Neighbor: first find the 5 most similar bottles in the train data and then determine from which farmer most of these 5 bottles are.

Note that WineData.pl contains 39 testdata_class/2 facts, one for each bottle in the test data. These facts tell us from which farmer the bottle really is. The first argument of such a fact is always the bottle-number; the second argument is the farmer. This information allows us to check how good the classifications are that we made before.

- Write a predicate that classifies all bottles in the test data and checks which classifications are correct (using the testdata_class/2 facts) and returns the accuracy of our classifications (accuracy is the number of correct classifications divided by the total number of classifications).
