Lunds Tekniska Högskola
Institutionen för datavetenskap

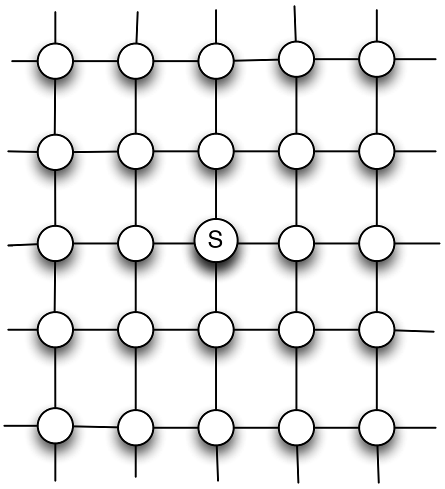
EDA132
VT 2017

Tillämpad Artificiell Intelligens
Applied Artificial Intelligence
Tentamen 2016–03–15, 14.00–19.00, MA:8

You can give your answers in English or Swedish.
You are welcome to use a combination of figures and text in your answers.
100 points, 50% needed for pass.

1 Search (JM): 11 points

Consider the unbounded regular 2D grid state space shown below. The start state is the origin (marked) and the goal state is (x, y).



1. (1) What is the branching factor b in this state space?
2. (1) How many distinct states are there at depth k (for $k > 0$)?
   (i) 4^k (ii) $4k$ (iii) k^4
3. (2) Breadth-first search *without* repeated-state checking *expands* at most
   (i) $((4^{x+y+1} - 1)/3) - 1$ (ii) $4(x + y) - 1$ (iii) $2(x + y)(x + y + 1) - 1$
   nodes before terminating.

4. (2) Breadth-first search *with* repeated-state checking *expands* up to (i) $((4^{x+y+1}-1)/3)-1$ (ii) $4(x+y)-1$ (iii) $2(x+y)(x+y+1)-1$ nodes before terminating.

5. (2) *True/False:* $h = |u-x| + |v-y|$ is an admissible heuristic for a state at (u, v).

6. (1) *True/False:* A^* search *with* repeated-state checking using h expands $O(x+y)$ nodes before terminating.

7. (1) *True/False:* h remains admissible if some links are removed.

8. (1) *True/False:* h remains admissible if some links are added between nonadjacent states.

2 Planning (JM): 14 points

A simple computer has a bunch of memory cells M (like $M_1, ..., M_n$) and some registers R (say, $R_1, ..., R_m$). Two computer instructions could be LOAD(M,R) (copy contents of M into R, overwriting what's there) and ADD(M,R) (add contents of M to contents of R, leaving result in R.)

1. (6) Express these instructions as STRIPS operators.

2. (4) With several registers, you could imagine a compiler being inefficient with register usage (overwriting partial results that could be reused, for instance). Relate what you know or can imagine about code optimization to what you know about planning.

3. (4) Would the planning graph representation be useful for this problem? Motivate how or why not?

3 Reasoning (JM): 1+7+7=15 points

Given the following can you conclude that the news about Sweden is a breaking news? Can your conclusion be reached using Backward Chaining? If yes, how? If no, why? Can your conclusion be reached using Resolution? If yes, how? If no, why?

News can either be fake or real, but never both.
Alternative news are not real.
News sent at night are alternative.
Fake news make it to the headlines.
Headline news are called breaking news.
Tonight a news came that Sweden has closed its borders to American citizens.

4 Unsupervised learning (PN): 30 points

In this question, you will implement an unsupervised learning algorithm: the bottom-up hierarchical clustering. The programming languages you can use for the implementation are: Python (preferred, if you know this language), Perl, Java, or Prolog. Please use only one of these programming languages unless agreed with your instructor. If you need a built-in function and you do not know exactly its name or parameters, invent something plausible.

4.1 Reading a Dataset and Converting the Nominal Attributes

Table 1 shows the "play tennis" dataset (Quinlan, 1986) with numeric and nominal attributes, where **Play** is the class and *Outlook*, *Temperature*, *Humidity*, and *Windy* are the attributes. The class will not be part of the computation.

Outlook	Temperature	Humidity	Windy	**Play**
sunny	85	85	FALSE	no
sunny	80	90	TRUE	no
overcast	83	86	FALSE	yes
rainy	70	96	FALSE	yes
rainy	68	80	FALSE	yes
rainy	65	70	TRUE	no
overcast	64	65	TRUE	yes
sunny	72	95	FALSE	no
sunny	69	70	FALSE	yes
rainy	75	80	FALSE	yes
sunny	75	70	TRUE	yes
overcast	72	90	TRUE	yes
overcast	81	75	FALSE	yes
rainy	71	91	TRUE	no

Table 1: The "play tennis" dataset with numeric attributes

4.1.1 Data Types

Linear classification algorithms and clustering only accept numerical data.

1. Describe how to convert the nominal (or categorical) attributes into numeric vectors. This process is called a one-hot encoding or a vectorizing.

2. Describe how you would apply this conversion to the data in Table 1. You will describe which attributes need to be converted; how you will

convert them to a vector; and you will convert manually the four first rows of the table (the numeric features will stay the same). You will consider Boolean values as categorical and you will ignore the class.

4.1.2 Programming

You will write a program to read a data set and vectorize it. The data set will consist of numeric and categorical values as in Table 1 and the output will be a numeric array.

To help the conversion, the data set will be associated with a structure variable, `struct_dataset`, describing it in terms of pairs: (name, type). The type is either `cat` for categorical, `num` for numeric, and `class` for the class. The data set in Table 1 has this structure:

```
struct_dataset = [('outlook', 'cat'), ('temperature', 'num'),
  ('humidity', 'num'), ('windy', 'cat'), ('play', 'class')]
```

You will split the program in five functions:

1. Write a function `read_tsv()` that reads the dataset in Table 1, where the columns are separated with tabulations. The input will be the file name and the output a list of lists, where the class will be excluded. For the two first rows in Table 1, you should have:

   ```
   [['sunny', '85', '85', 'FALSE'],
   ['sunny', '80', '90', 'TRUE'], ...
   ```

2. Write a function `make_dict()` that converts the data set into dictionaries (maps in Java). The input will be list of lists from the previous function and the `struct_dataset` variable. The output will be a list of rows, where each row is a dictionary. The key will be the column name and the value, the value of the attribute.

   For the two first rows in Table 1, you should have:

   ```
   [{'outlook': 'sunny', 'temperature': '85', 'humidity': '85',
     'windy': 'FALSE'},
   {'outlook': 'sunny', 'temperature': '80', 'humidity': '90',
     'windy': 'TRUE'}, ...
   ```

   Your function will use the `struct_dataset` variable to name the keys;

3. Write a function `collect_values()` that takes the list of dictionaries above and the `struct_dataset` variable as input and collects all the values of a given key if it corresponds to a categorical variable. You will sort the values and you will store the results in a dictionary. For Table 1, this should be:

```
{'outlook': ['overcast', 'rainy', 'sunny'],
'windy': ['FALSE', 'TRUE']}
```

4. Write a function `one_hot_vector()` that takes a value and an ordered list of values as input and maps the value to its index in a list. The function will then convert this index in a one-hot vector: A vector of 0's except one value corresponding to the index that is set to one. For the outlook attribute above, you should have these conversions:

   ```
   one_hot_vector('overcast',['overcast', 'rainy', 'sunny'])
     --> [1, 0, 0]
   one_hot_vector('rainy', ['overcast', 'rainy', 'sunny'])
     --> [0, 1, 0]
   one_hot_vector('sunny', ['overcast', 'rainy', 'sunny'])
     --> [0, 0, 1]
   ```

5. Write a function `vectorize()` that converts the data set into numeric vectors. Your function will use the functions above and the `struct_dataset` variable to carry out the conversion into the right type. Each row (vector) will be represented as a list and the table as a list of lists. The three first rows should look like this:

   ```
   [[0, 0, 1, 85.0, 85.0, 1, 0],
   [0, 0, 1, 80.0, 90.0, 0, 1],
   [1, 0, 0, 83.0, 86.0, 1, 0],
   ...
   ```

4.2 Hierarchical Clustering

The hierarchical clustering algorithm is an unsupervised classifier that creates a partition from a dataset. Let us suppose that the data set consists of N observations. In the beginning, all the points are individual clusters, i.e., we have N clusters. The first iteration of the algorithm finds the two closest points and merges them into a new cluster consisting of the list of the two points. The process is started again with $N - 1$ clusters until we reach the number of clusters we want.

4.2.1 Manual Computations

1. Given the three points:

   ```
   [[0, 0, 1, 85.0, 85.0, 1, 0],
   [0, 0, 1, 80.0, 90.0, 0, 1],
   [1, 0, 0, 83.0, 86.0, 1, 0]]
   ```

compute the Euclidian distance between the first and second points, first and third, and second and third. You do not need to evaluate the square roots;

2. What are the two closest points?
3. You will merge the two closest points into a list and compute the distance between the resulting list and the remaining point. There two ways to do this, where the distance can be:

   - the distance between the remaining point and the point in the list that is the farthest. This distance is called max;
   - the distance between the remaining point and the point in the list that is the closest. This distance is called min.

   Give the two values of these distances;
4. Try to outline the properties of these two distances.

4.2.2 Programming Hierarchical Clustering

You will now program the hierarchical clustering algorithm. Be sure that your program works for any number of clusters as well as with any number of observations of any dimensions. Please, use the variable k for the number of clusters, q for the number of observations, and N for the dimension of each observation. In Table 1, after the vectorization, we have $q = 14$ and $N = 7$. Your program should run for any value of q and N, for instance $q = 3000$ and $N = 10$.

1. Write a function `euclidian_distance()` that computes the Euclidian distance between two points. The coordinates of a point are given in a list, for instance:

   ```
   [0, 0, 1, 85.0, 85.0, 1, 0]
   ```

2. Write a function `cluster_distance()` that computes the distance between two clusters. Each cluster is represented as a list of lists, for instance:

   ```
   cluster1 = [[0, 0, 1, 85.0, 85.0, 1, 0],
       [0, 0, 1, 80.0, 90.0, 0, 1]]
   ```

   and

   ```
   cluster2 = [[1, 0, 0, 83.0, 86.0, 1, 0]]
   ```

You will use the `min` distance.

3. Write a `closest_clusters()` function that determines the two closest clusters. As input the clusters will be represented as lists of points, and the partition as a list of clusters. For instance, the list below:

   ```
   [
       [[0, 0, 1, 85.0, 85.0, 1, 0],
       [0, 0, 1, 80.0, 90.0, 0, 1]],
       [[1, 0, 0, 83.0, 86.0, 1, 0]]
   ]
   ```

   represents two clusters, the first one with two points and the second, with one point. The function will return the indices of the two closest clusters.

4. Write a `merge_clusters()` function that merges two clusters in a partition. The input will be the indices of the two clusters and the current partition. The output will be the new partition.

Finally, using the functions above, write the program that carries out a hierarchical clustering of the data set in Table 1. You can choose $k = 2$

5 Probabilistic Reasoning / Learning (EAT): 4+12+4+10 = 30 points

Assume you have a new vacuum cleaner robot that is equipped with a high-resolution camera, so that it theoretically could distinguish dirt (gravel, bread crumbs, peas, pieces of cucumber - whatever your kids manage to spread around the kitchen table) from small toy pieces (mostly Lego bricks), which could save you endless hours of trying to find Lego Star Wars Darth Vader's laser sword in the dirt bag of the vacuum cleaner after it accidentally got cleaned up. Problem is, you need to train the respective classification software to work in your house...
The robot has software that does some background segmentation on the camera images, and it can also find features of objects segmented from the background, like colour, size, shape or texture. You decide to start out with some rather simple items to get a feeling for how this works, so you want to try "classic" 2x4, 2x2, and 1x1 Lego pieces (L) against pieces of cucumber and peas (D). Obviously, the different properties of these items are not really connected to each other when you look only at one group of items.
You have the robot take pictures of 100 items according to this choice, and the software gives you the counts for the following features: colour (C, "green" or "not green"), shape (S, "block" or "not block"), and texture (T,

"shiny" or "not shiny"). Now you are all set and can use the data to train a classifier, you think, and then you figure that you forgot to actually label the items in the software, you only remember that you had about 30% of Lego pieces in the collection of items.

At least you have the following counts for combinations of attributes (features):

	C = green		C ≠ green	
	S ≠ block	S = block	S ≠ block	S = block
T ≠ shiny	35	7	10	15
T = shiny	5	11	8	9

a) Draw the (optimal) Bayesian network that represents your problem. What assumption(s) can you make to keep things simple? What type of model is this, actually?

b) Which approach for training would you have been able to use if you had labeled the items properly? Which algorithm can you use now, given that you do not have the labels? Describe this second algorithm and show the starting point for the first step (i.e., show what to do with the numbers you have).

c) You find your results quite ok, however, you have in the meantime found a model that describes exactly your problem, but has been trained with a lot more examples, which you consider more reliable, and you decide to use that one instead of your own for testing the robot. This model gives you these numbers:

$$P(C = green|D) = P(S \neq block|D) = P(T \neq shiny|D) = 0.6,$$

$$P(C \neq green|L) = P(S = block|L) = P(T = shiny|L) = 0.8,$$

$P(D) = 0.73$ and $P(L) = 0.27$. What would the robot report if it finds something with the following properties: *green, not block, and shiny* according to a Naïve Bayes Classifier and the model above?

d) Explain the following terms and their potential connections in the context of probabilistic reasoning and learning: *Classification*, *Markov* *assumption*, *Learning*, *Prediction*, *Bayesian network*, *Tracking*, *MAP-hypothesis*, *Optimal Bayes Learner*, *Stationary process*, *Maximum-Likelihood hypothesis*.

References

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

Good Luck!