Lunds Tekniska Högskola
Institutionen för datavetenskap

EDA132
VT 2015

Tillämpad Artificiell Intelligens
Applied Artificial Intelligence
Tentamen 2015–03–17, 08.00–13.00, Sparta A,B

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

1 Search: (12 points)

Bilbo and his companions have reached the mountain where Smaug, the dragon, holds its treasure. They enter the mountain's internal labirynth via a large cave (marked S in the figure 1) and discover a system of caves connected by corridors. They start exploring the caves, looking for ones holding gold (G1 and G2, it happens that there are two of them, but Bilbo and his companions needs not to be aware of this fact).

Each corridor takes some hours to pass (denoted by numbers by the arrows describing direction in which a corridor might be taken). Each cave holds a sign saying " you still need at least x hours to reach treasure".

Luckily, Bilbo has read once an old manuscript describing strategies of exploring labirynths and plans to use his knowledge now. For each of the following search strategies, indicate which gold cave is reached (if any) and list, in order, all the caves visited in between (i.e., states popped off of the OPEN/FRINGE list. When all else is equal, caves should be explored in alphabetical order.

You may use the form below or copy its structure on an answer sheet. Remember that some nodes may occur on this list more than once!

Breadth-First (2p) Gold cave reached: _____

States popped off OPEN:_____

Best-First (Greedy) using $f = h$

(2p) Gold cave reached: _____

States popped off OPEN:_____

Iterative Deepening (2p) Gold cave reached: _____

States popped off OPEN:_____

A* (3p) Gold cave reached: _____

States popped off OPEN:_____

— (1p) Is the heuristic estimate illustrated in Fig. 1 admissible? Why or why not?

— (1p) Is the heuristic estimate illustrated in Fig. 2 admissible? Why or why not?

— (1p) Modify any of the heuristic estimates shown in Figs 1 and 2, so that the A* search is optimal, i.e. it expands the least number of nodes searching for the optimal path.

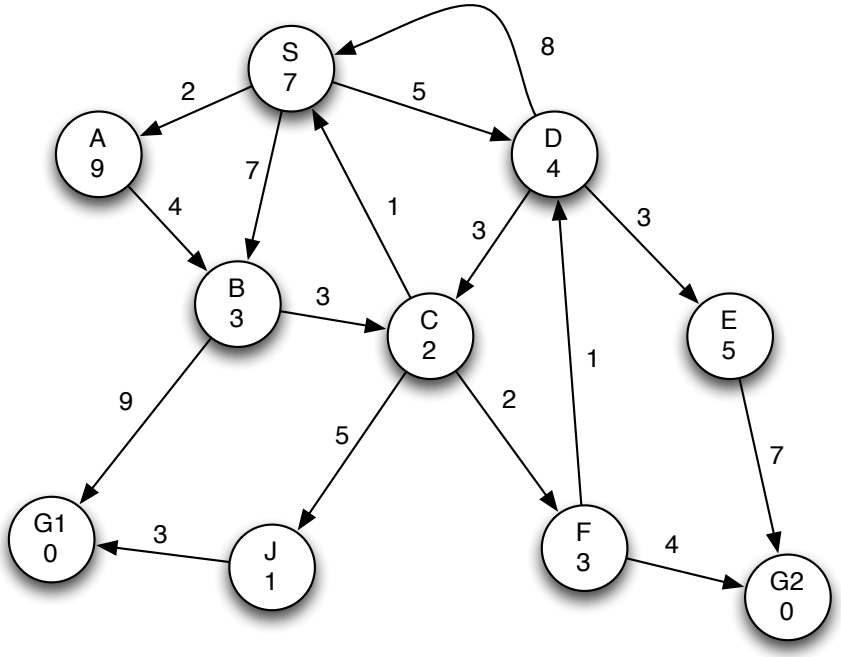


Figure 1: A search space, with costs (over arcs) and heuristic estimates (in nodes).

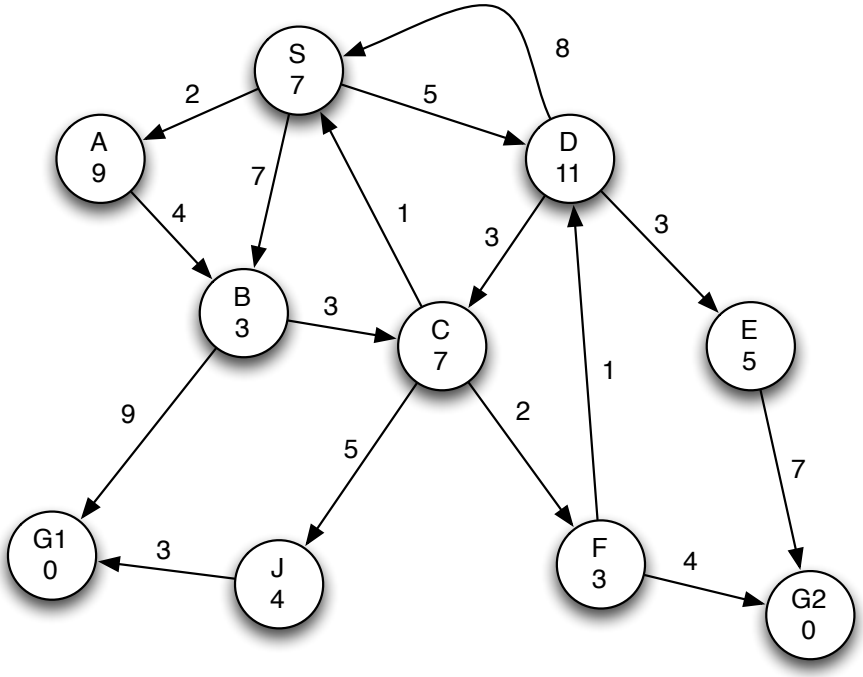


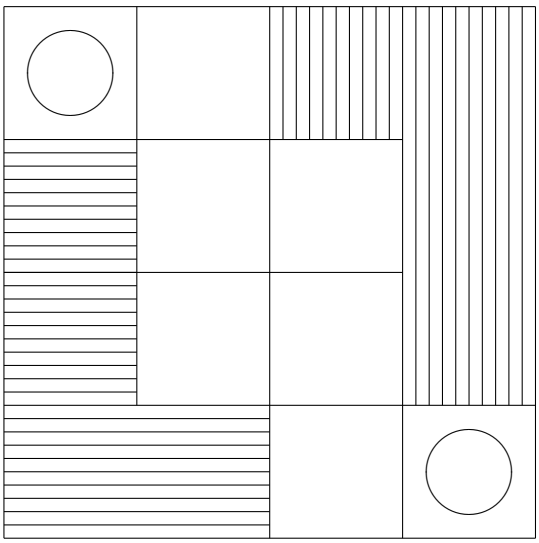
Figure 2: A search space, with costs (over arcs) and heuristic estimates (in nodes).

2 Games: (12 points)

L-game is played by two players on a 4×4 board. Each player has an L-shaped figure which can be turned or rotated in all directions. A move consists of two parts:

1. The player lifts her L-figure and puts it down on the board in a different position than before.
2. If she wants, she can also move one of the two neutral pieces[1] to a new position.

The player who cannot move her L to a new position, looses the game. The following picture illustrates the board in the beginning of the game.



(6p) Assume now that you want to write a program that could play the L-game. Your task consists of representing the problem as a *search* problem. Describe how state and operators (possible moves) could be represented, how the goal state will be recognized (a *goal-test* function), how possible moves will be generated (a *successor* function), how one of the possible moves will be chosen (a *choose-move* function), etc.

In order to avoid misunderstanding, some precision is necessary in your answer. Therefore it would be a benefit if you could use e.g. list structures (or whatever you deem appropriate) to define the necessary data types you choose to represent state and operators. You can write your functions using a pseudocode sufficiently close to a programming language (it is advised to choose one that I know:-).

[1] A neutral piece, marked by a circle on top of it, covers just one square of the board.

Remember that your program is going to play against an opponent. Therefore during the search for the next best move you should take into account the possible move of the opponent.

(3p) Assume that the search space for the L-game is huge (which is not true in reality). In such case you would probably like to use some form of search-tree cutoff in order to minimize the search time. Given your original solution, how would you introduce α-β pruning in it? I would appreciate an illustrative example using the L-game (NOT the one from the book!).

(3p) Finally, let us look at the α-β pruning. In both α-cutoff and β-cutoff we cut some part of the search tree because we conclude that there is no use in expanding this particular branch of the tree. We perform same reasoning in both α and β cases. Therefore there is no difference between them! Please convince me that α-cutoff and β-cutoff differ from each other (that is, that the preceding sentence is false). The best way of doing this is to describe HOW do they differ, I suppose.

3 Reasoning: (10 points)

1. *"Heads I win; tails you lose"* can be represented in propositional logic in the following way:

   $Heads \rightarrow WinMe$
   $Tails \rightarrow LoseYou$
   $\neg Heads \rightarrow Tails$
   $LoseYou \rightarrow WinMe$

   - (2p) Is it possible to use the inference method *Forward Chaining* to derive $WinME$? Motivate your answer.
   - (2p) Is it possible to use the inference method *Backward Chaining* to derive $WinME$? Motivate your answer.
   - (2p) Is it possible to use resolution to derive $WinMe$? Motivate your answer.

2. Mark the following formulae as one of:

   - satisfiable but not valid
   - valid
   - not satisfiable

   assuming interpretations belong to the specified set:

   (a) (2p) $\neg(\neg a \vee b) \wedge (\neg b \wedge \neg a)$
   Interpretations: all possible (assignments for a and b).

(b) (2p) $(\neg a \wedge b) \vee (a \wedge \neg b)$
Interpretations: $\{< a = \text{False}, b = \text{True} >, < a = \text{True}, b = \text{False} >\}$.

4 MAP-hypothesis: (10 points)

You are really fascinated by the classic issue of finding a proof for P=NP (or the respective negation). The usual comment you seem to find about this issue is "Yeah, when pigs fly ...!". From those comments you conclude that the probability for observing a flying pig in case it is possible to find a proof is 0.8, but only 0.1 if there is not. This morning you were still pretty sure that no one could ever find a suitable proof and determined the probability for this happening as 0.1. You step out your door and observe a pig flying by. Should you start believing that it is possible to find a proof? Motivate your answer! If not, how many flying pigs do you need to observe until you start revising your initial understanding when you reason based on a MAP-approach?

5 HMM, Tracking (Smoothing): (20 points)

You are somewhat disappointed by the weather forecast you get to read in the local newspaper and have decided to develop your own model, specifically you settle for a hidden Markov model (HMM). Mostly you are interested in knowing whether it is going to rain during a day that starts out sunny, so that you can take your rain clothes with you in the morning, but do not have to carry them all the time. Before relying on the predictions your model generates, you test the smoothing results it provides against a real sequence of weather changes.
You are interested in the variable *WeatherType*, that can have the values *rain* (for persistent rain during a day), *dry* (for persistently dry weather during a day), and *unstable*, which gives you 3 states to handle. Your observations (when you stick your nose out your door) are a combination of current weather (rain, dry) and wind (weak, strong), thus you have four possible values for your sensor model.

The independent parts of your transition model are:
P(rain | rain) = 0.65, P(unstable | rain) = 0.1
P(rain | unstable) = 0.4, P(dry | unstable) = 0.25
P(unstable | dry) = 0.45, P(dry | dry) = 0.35

and your observations seem to follow this pattern:
P(strong and (rain or dry) | unstable) = 0.4
P(weak and (rain or dry) | unstable) = 0.1

P(strong and rain | rain) = 0.35
P(weak and rain | rain) = 0.55
P((weak or strong) and dry | rain) = 0.05
P(strong and dry | dry) = 0.4
P(weak and dry | dry) = 0.5
P((weak or strong) and rain | dry) = 0.05

You start out with a prior distribution that favours unstable weather (0.5) (as it is April), while persistent weather conditions are equally likely for both rain and dry weather.

a) Set up and explain the necessary matrices for HMM-based forward-backward calculations.

b) Assume an observation sequence of ("weak & rain"), ("strong & dry"), ("strong & dry"), ("weak & rain") for the first four days covered by your data. Calculate the smoothed estimates for day 2 (based on the first three observations) and 3 (based on all four observations.

c) The actual sequence of weather observations for these four days was "rain", "unstable", "unstable", "rain". Would you say your model works satisfyingly? Does a "hit", i.e., a correct estimate, mean that you are "safe" from that point on?

d) One of the assignments asked for HMM-based robot localisation in a (fixed-size) grid world. Would it make sense to use an HMM for a more general case of a localisation problem? Why? Are there other (similar) approaches and which ones (and how do they compare to the HMM-approach)?

e) What must be fulfilled for a process to be compliant with the Markov-assumption? Is the general case of robot localisation still a Markov process?

6 Logistic regression: (12 points)

Logistic regression is a classification technique based on the logistic function.

1. (a) Write this logistic function in an n-dimensional space: $y = Logistic(\mathbf{x})$; You will use the notation $\mathbf{w}$ to define the weight vector;
   (b) Rewrite the logistic function in a two-dimensional space: $y = Logistic(x)$ with the weight vector (w_0, w_1).
   (c) Draw this function;

(d) Rewrite the logistic function in a tree-dimensional space:

$$y = Logistic(x_1, x_2)$$

with the weight vector (w_0, w_1, w_2);

2. Table 1 shows an (invented) set of six individuals. Represent graphically this data set. You will use dots for class 1 and crosses for class 0.

3. We want to use logistic regression to discriminate between vipers (Huggormar) and grass snakes (snok). You will assess the classification accuracy of two different $\mathbf{w}$ vectors:

   - $\mathbf{w}_1 = (50, -0.10, -0.50)$ and
   - $\mathbf{w}_2 = (100, -0.30, -0.30)$.

   (a) Write the formula giving the probability of an individual snake $\mathbf{x}$ to belong to class 1, $P(1|\mathbf{x})$, when the weight vector is $\mathbf{w}$;

   (b) Ideally, what should be the values of these probabilities for the first and 6th snakes in Table 1. Please write one value for each snake: $P(1|\mathbf{x}_1)$ and $P(1|\mathbf{x}_6)$;

   (c) Compute this probability for the first and 6th snakes in the table with the two vectors $\mathbf{w}_1$ and $\mathbf{w}_2$. You have to use two formulas and produce two numbers for each snake: $P_{\mathbf{w}_1}(1|\mathbf{x}_1)$, $P_{\mathbf{w}_1}(1|\mathbf{x}_6)$, $P_{\mathbf{w}_2}(1|\mathbf{x}_1)$, and $P_{\mathbf{w}_2}(1|\mathbf{x}_6)$.

   (d) Which vector seems to best predict the classes?

   (e) Using a product of probabilities, write the equation that defines the optimal weight vector for the whole data set. You will not try to compute it or determine the optimal value of $\mathbf{w}$.

4. The optimal weight vector found by the R statistical package is: (71.2407, -0.1823, -0.3198). Using this vector, compute the probability of the second and 9th snakes to belong to class 1, $P(1|\mathbf{x}_2)$ and $P(1|\mathbf{x}_9)$. Is this value surprising?

7 Decision trees: (12 points)

ID3 is a machine-learning algorithm designed for categorical data. In this question, you will examine how ID3 can be adapted to numerical attributes.

1. ID3 induces decision trees from data sets and uses the concept of entropy to evaluate the partition of a set.

#	Length (cm)	Weight (g)	Type	Class
1.	105	240	grass snake	0
2.	75	180	grass snake	0
3.	92	230	grass snake	0
4.	100	240	grass snake	0
5.	110	250	grass snake	0
6.	55	100	viper	1
7.	60	110	viper	1
8.	50	90	viper	1
9.	75	180	viper	1
10.	60	120	viper	1

Table 1: The data set

(a) Write the definition of the entropy of a binary partition consisting of P positive examples, and N negative ones.

(b) A categorical attribute with v values would split the initial set into v subsets. Represent this process graphically and write the definition of the information gain resulting from this split.

2. ID3 and numerical attributes.

(a) Should you apply the original information gain to the data set in Table 1, what would be the result? Give the number of subsets each attribute would produce.

(b) Instead, many tree induction algorithms partition the set using binary splits. Implement such a procedure following this outline:

- Given an attribute with numerical values, you will first sort the values in increasing order and consider each value as a potential split point.
- You will scan the values of the attribute and determine the optimal split point according to a criterion that you will imagine.

(c) Write an information gain procedure so that it can handle and compare numerical and categorical attributes.

8 Language Technology: (12 points)

Figure 3 shows a word cloud that visualizes the importance of words in a collection of documents. More important words, like *algoritmer* or *projekt*, use a larger font size than apparently less important ones like *distribuerade*

or *teorin*. In Fig. 3, the collection of documents consists of the web pages of the Institute of computer science at LTH (Institutionen för datavetenskap).

In this exercise, you will write a program to compute the weights of the words in a collection of documents so that you can scale the font size with the word importance. As language, you can choose among Java, Python, Perl, etc., provided that the language has the hash map data structure (dictionaries in Python).

Figure 3: A word cloud extracted from the web pages of the Institute of computer science at LTH. Courtesy of Jonas Wisbrant

1. Indexing

   As input, you will be given a corpus: A collection of N documents with the names: file1.txt, file2.txt, file3.txt, ..., fileN.txt containing words separated by white spaces. The lines below show an example of two such documents:

   file1.txt: chrysler plans new investments in latin america and in canada

   file2.txt: chrysler plans major investments in mexico

   In the first step, you will build indexes of all the words contained in all the documents.

   (a) Write a program that reads one document and outputs an index file: **file_name.idx**. The index file will contain the list of the words in the document, one word per line, where each word is

followed by the list of its positions in the document. The positions will correspond to the number of words from the beginning and will be separated by spaces.

Table 2 shows the two index documents resulting from the two text documents above. Note that the word *in* shows twice in file1.txt and hence in file1.idx with the positions 5 and 9.

file1.idx	file2.idx
and 8	chrysler 1
america 7	in 5
canada 10	investments 4
chrysler 1	major 3
in 5 9	mexico 6
investments 4	plans 2
latin 6	
new 3	
plans 2	

Table 2: Index files

(b) Write a program that reads all the documents of the corpus and outputs as many index files.

2. Scoring

You will score the words in the corpus using the $TF \cdot IDF$ function.

(a) TF is the term frequency, the number of times it occurs in the corpus. Write the program that reads the index files and computes the frequency of all the words in the collection. In Table 2, *in* has a count of 3, while *new* has a count of 1. Write a program that reads all the index files and computes the term frequency of each word.

(b) IDF is the inverse document frequency: $\frac{1}{DF}$, where DF is the number of documents that contain the word. For instance, $DF(\texttt{in}) = 2$, $DF(\texttt{chrysler}) = 2$, and $DF(\texttt{new}) = 1$. Write a program that reads all the index files and computes the inverse document frequency of each word.

(c) Finally, combine your programs to output $TF \cdot IDF$ for each word.

Good Luck!