COMP3308/3608 Introduction to Artificial Intelligenece
(regular and advanced)

semester 1, 2025

Information about the exam

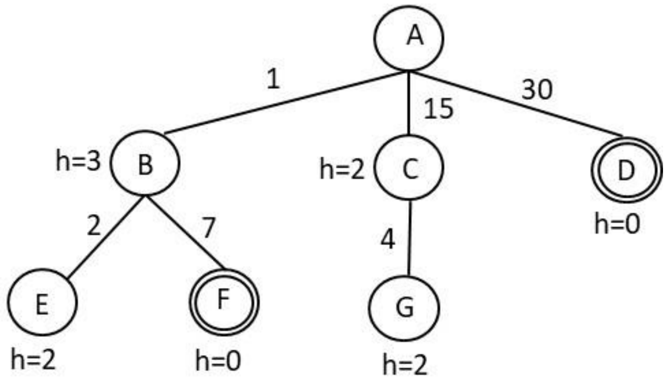
- The exam is in-person, paper-based
- Permitted materials:
    - One sheet of your own notes - double-sided A4 size, typed or handwritten
    - Calculator - handheld, non-programmable
- No other materials or devices are allowed
- You will need to leave the A4-sheet with the exam paper, so it is recommended that you make a copy of it for your own records
- You need to have a photo identification - Student ID card or other accepted documents
- The exam paper is like a booklet. You write your answers on the exam paper, in the space provided. There is a question, then a space for you to write the answer; another question and space for the answer, etc. To write the answers you should use black or blue pen, not pencil.

- The exam paper is **confidential**. You must not discuss the exam questions with other people, post or distribute the exam questions in any way **after the exam**.
- The duration of the exam is standard: 2 hours + 10 minutes reading time.
- The exam is worth 100 marks (=60% of your final mark). To pass the course you need at least 40% on the exam (i.e. 40 marks), regardless of what your mark during the semester is. This is a school rule.
- All material is examinable except week 1, week 13a (recommender systems), historical context, Matlab and Weka.
- There are 3 types of questions: 1) questions requiring short answers, 2) problem-solving questions and 3) multiple-choice questions (small number).

Sample exam questions

In addition to these questions please also see on Canvas:
Search: Jessica's Weeks_2-3_Practice.pdf (prepared by Jessica McBroom)
Bayesian networks: BN_practice_questions.pdf (prepared by Jessica McBroom)

Question 1. (Type 2 – problem solving)
In the tree below the step costs are shown along the edges and the h values are shown next to each node. The goal nodes are double-circled: F and D.



Write down the order in which nodes are expanded using:

a) Breadth-first search
b) Depth-first search
c) Uniform cost search
d) Iterative deepening search
e) Greedy search
f) A*

In case of ties, expand the nodes in alphabetical order.

Answer:

a) Breadth-first search: ABCD
b) Depth-first search: ABEF
c) Uniform cost search: ABEF
d) Iterative deepening search:AABCD
e) Greedy search: AD
f) A*: ABEF

Review:

- BFS: Expands the shallowest unexpanded node
- DFS: Expands the deepest unexpanded node

- UCS: Expands the node with the smallest path cost *g(n)*
- IDS: DFS at levels *l* = 0, 1, 2, etc. Expands the deepest unexpanded node within level *l*
- Greedy: Expands the node with the smallest heuristic value *h(n)*
- A*: Expands the node with the smallest *f(n)=g(n)+h(n)*

Question 2. (Type 1 – short answers)

Answer briefly and concisely:

a) A* uses admissible heuristics. What happens if we use a non-admissible one? Is it still useful to use A* with a non-admissible heuristic?

b) What is the advantage of choosing a dominant heuristic in A* search?

c) What is the main advantage of hill climbing search over A* search?

Answers:

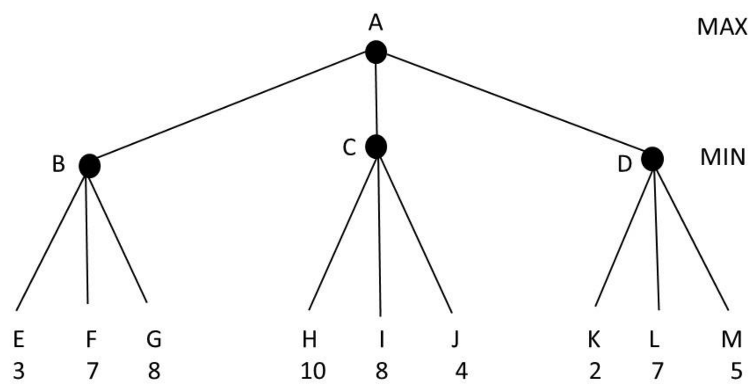
a) Not optimal anymore. But it could still find a reasonably good solution in acceptable time, depending on how good the heuristic is.

b) Fewer nodes expanded. As a result, the optimal solution will be found quicker.

c) Space complexity – keeps only the current node in memory.

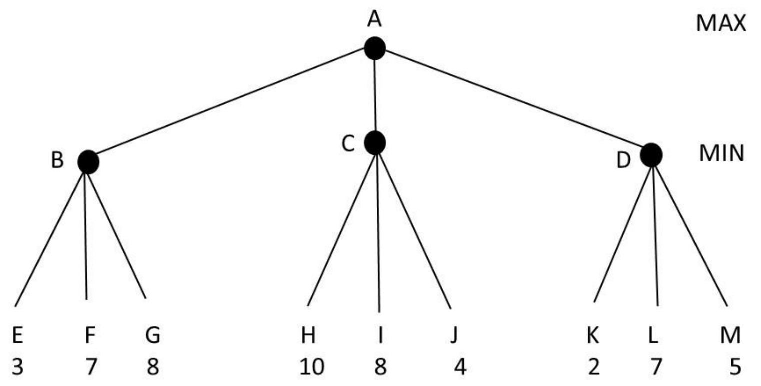
Question 3. (Type 2 – problem solving)

Consider the following game in which the evaluation function values for player MAX are shown at the leaf nodes. MAX is the maximizing player and MIN is the minimizing player. The first player is MAX.



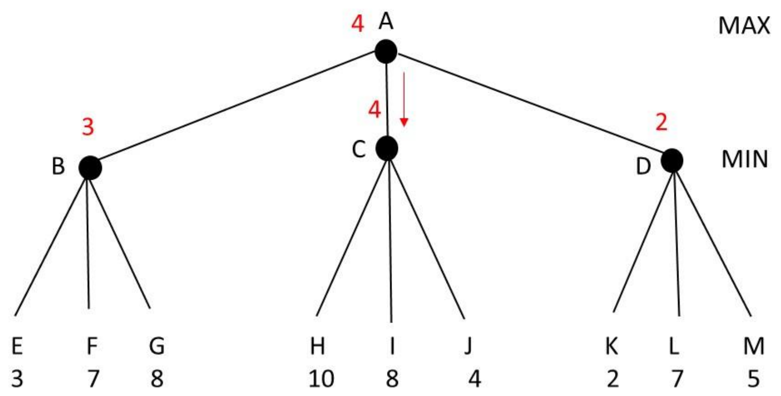
a) In the figure above, write the backed-up values that are computed by the **minimax** algorithm for all nodes. Show with an arrow the move that MAX should choose.

b) We now use the **alpha-beta** algorithm. In the figure below, cross all the branches that would be pruned, e.g. AB etc. Assume that children are visited left-to-right (as usual).

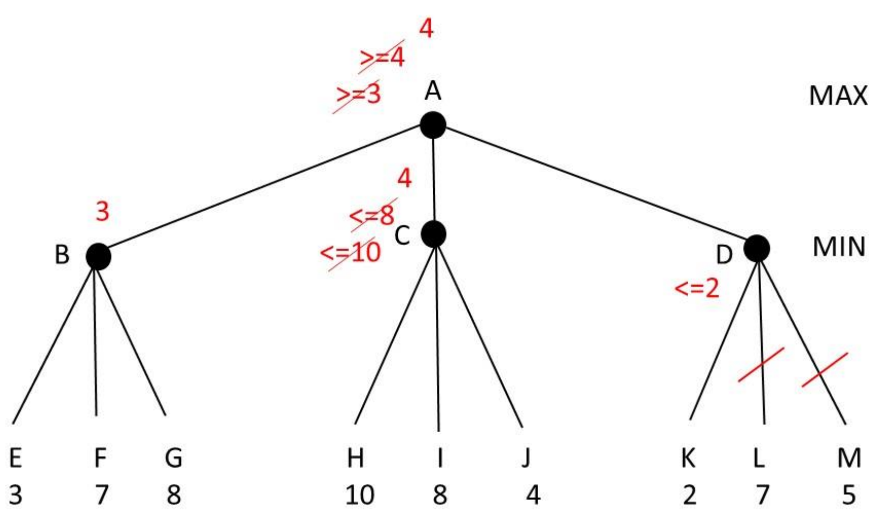


Solution:

a) The values are as shown and MAX should chosse the move to C:



b) LD and MD will be pruned:



Question 4. (Type 1 – short answers)

Answer briefly and concisely:

a) The 1R algorithm generates a set of rules. What do these rules test?

b) *Gain ratio* is a modification of *Gain* used in decision trees. What is its advantage?

c) Propose two strategies for dealing with missing attribute values in learning algorithms.

d) Why do we need to normalize the attribute values in the k-nearest-neighbor algorithm?

e) What is the main limitation of the perceptrons?

f) Describe an early stopping method used in the backpropagation algorithm to prevent overfitting.

g) The problem of finding a decision boundary in support vector machine can be formulated as an optimisation problem using Lagrange multipliers. What are we maximizing?

h) In linear support vector machines, we use dot products both during training and during classification of a new example. What vectors are these products of?

   During training:

   During classification of a new example:

Answers:

a) They test the values of a single attribute.

b) It penalizes highly-branching attributes by taking into account the number and the size of branches.

c) Strategy 1: Use the attribute mean to fill in the missing value.
   Strategy 2: Use the attribute mean for all examples belonging to the same class.

d) As different attributes are measured on different scales, without normalization the effect of the attributes with smaller scale of measurement will be less significant than those with larger.

e) Can separate only linearly separable data.

f) Available data is divided into 3 subsets:
   Training set – used for updating the weights.
   Validation set – used for early stopping.
   Training is stopped when the error on the validation set increases for a pre-specified number of iterations.

g) The margin of the hyperplane.

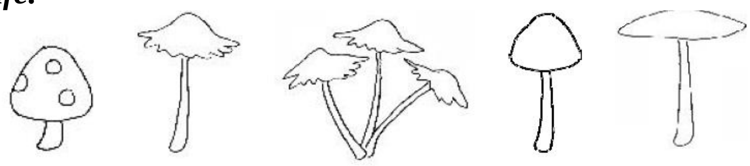
h) During training: Pairs of training examples.
   During classification of a new example: The new example and the support vectors.

Question 5. (Type 2 – problem solving)

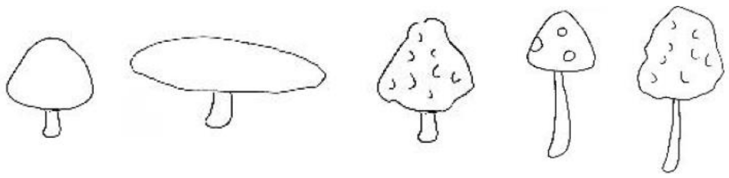
Consider the task of learning to classify mushrooms as ***safe*** or ***poisonous*** based on the following four features: stem = {short, long}, bell = {rounded, flat}, texture = {plain, spots, bumpy, ruffles} and number = {single, multiple}.

The training data consists of the following 10 examples:

Safe:



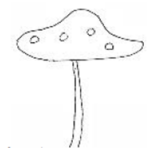
Poisonous:



These examples are also shown in the table below:

n	stem	bell	texture	number	class
1	short	rounded	spots	single	safe
2	long	flat	ruffles	single	safe
3	long	flat	ruffles	multiple	safe
4	long	rounded	plain	single	safe
5	long	flat	plain	single	safe
6	short	rounded	plain	single	poisonous
7	short	flat	plain	single	poisonous
8	short	rounded	bumpy	single	poisonous
9	long	rounded	spots	single	poisonous
10	long	rounded	bumpy	single	poisonous

a) Use Naïve Bayes to predict the class of the following new example: stem=long, bell=flat, texture=spots, number=single. Show your calculations.



a) How would 3-Nearest Neighbor using the Hamming distance classify the same example as above? Explain your answer. (Hint: The Hamming distance is the number of different feature values).

b) Consider building a decision tree. Calculate the information gain for ***texture*** and ***number*** and briefly show your calculations Which one of these two features will be selected and why?

You may use this table:

x	y	-(x/y)*log$_2$(x/y)	x	y	-(x/y)*log$_2$(x/y	x	y	-(x/y)*log$_2$(x/y	x	y	-(x/y)*log$_2$(x/y
1	2	0.50	4	5	0.26	6	7	0.19	5	9	0.47
1	3	0.53	1	6	0.43	1	8	0.38	7	9	0.28
2	3	0.39	5	6	0.22	3	8	0.53	8	9	0.15
1	4	0.5	1	7	0.40	5	8	0.42	1	10	0.33
3	4	0.31	2	7	0.52	7	8	0.17	3	10	0.52
1	5	0.46	3	7	0.52	1	9	0.35	7	10	0.36
2	5	0.53	4	7	0.46	2	9	0.48	9	10	0.14
3	5	0.44	5	7	0.35	4	9	0.52			

d) Consider a single perceptron for this task. What is the number of inputs? What is the dimensionality of the weight space? Briefly explain your answer.

Solution:

a) Naïve Bayes:
The new example is the evidence E.
E1=stem=long, E2=bell=flat, E3=texture=spots, E4=number=single

We need to compute P(safe|E) and P(poisonous|E) and compare them.

$$P(safe \mid E) = \frac{P(E1 \mid safe)\, P(E2 \mid safe)\, P(E3 \mid safe) P(E4 \mid safe) P(safe)}{P(E)}$$

P(safe)=5/10=1/2
P(E1|safe)=P(stem=long|safe)=4/5
P(E2|safe)=P(bell=flat|safe)=3/5
P(E3|safe)=P(texture=spots|safe)=1/5
P(E4|safe)=P(number=single|safe)=4/5

P(poisonous)=5/10=1/2
P(E1|poisonous)=P(stem=long|poisonous)=2/5
P(E2|poisonous)=P(bell=flat|poisonous)=1/5
P(E3|poisonous)=P(texture=spots|poisonous)=1/5
P(E4|poisonous)=P(number=single|poisonous)=5/5

$$P(safe \mid E) = \frac{\frac{4}{5}\frac{3}{5}\frac{1}{5}\frac{4}{5}\frac{1}{2}}{P(E)} = \frac{\frac{24}{625}}{P(E)} \qquad P(poisonous \mid E) = \frac{\frac{2}{5}\frac{1}{5}\frac{1}{5}\frac{5}{5}\frac{1}{2}}{P(E)} = \frac{\frac{5}{625}}{P(E)}$$

=> Naïve Bayes predicts ***safe***.

b) The three nearest neighbors have a distance of 1 and are examples 2 (safe), 5 (safe) and 9 (poisonous). The majority vote is safe, hence the new example will be classifed as safe.

c) Decision tree

H(S)=I(5/10,5/10)= 1 bit

Split on ***texture***:
H(S_{spots})=I(1/2,1/2)=1 bit
H($S_{ruffles}$)=I(2/2,0/2)=0 bits
H(S_{plain})=I(2/4,2/4)=1 bit

$H(S_{bumpy})=I(0/2,2/2)=0$ bits
$H(S|texture)=2/10*1 + 2/10*0 + 4/10*1 + 2/10*0=6/10=0.6$ bits
gain(texture)=1-0.6=0.4 bits

Split on ***number***:
$H(S_{single})=I(4/9,5/9)=-4/9\log 4/9-5/9\log 5/9=0.52+0.47=0.99$ bits
$H(S_{multiple})=I(1/1,0/1)=0$ bits
$H(S|number)=9/10*0.99 + 1/10*0=0.891$ bits
gain(number)=1-0.891=0.109 bits

gain(texture) > gain(number) => ***texture*** will be selected as it has a higher information gain

d) There should be 10 inputs, 1 for each attribute value. This means using binary encoding of the attributes and their values (e.g. 10 for ***stem=short*** and 01 for ***stem=long***). Binary encoding is the most popular encoding for nominal attributes.

Hence, the weight space will have a dimensionality of 11 (10 weights +1 bias weight).

Question 6. (Type 2 – problem solving)

Given the training data in the table below where *credit history*, *debt*, *collateral* and *income* are attributes and *risk* is the class, predict the class of the following new example using the 1R algorithm: *credit history=unknown, debt=low, collateral=none, income=15-35K*. Show your calculations.

credit history	debt	collateral	income	risk
bad	high	none	0-15k	high
unknown	high	none	15-35k	high
unknown	low	none	15-35k	moderate
unknown	low	none	0-15k	high
unknown	low	none	over 35k	low
unknown	low	adequate	over 35k	low
bad	low	none	0-15k	high
bad	low	adequate	over 35k	moderate
good	low	none	over 35k	low
good	high	adequate	over 35k	low
good	high	none	0-15k	high
good	high	none	15-35k	moderate
good	high	none	over 35k	low
bad	high	none	15-35k	high

Solution:
1. Attribute *credit history*
bad:0 low, 1 moderate, <u>3 high</u> => risk=high, errors: 1/4
unknown: <u>2 low</u>, 1 moderate, 2 high => risk=low, errors: 3/5
good: <u>3 low</u>, 1 moderate, 1 high => risk=low, errors: 2/5
total errors: 6/14

2. Attribute *debt*
high: 2 low, 1 moderate, <u>4 high</u> => risk=high, errors: 3/7
low: <u>3 low</u>, 2 moderate, 2 high => risk=low, errors: 4/7
total errors: 7/14

3. Attribute *collateral*
none: 3 low, 2 moderate, <u>6 high</u> => risk=high, errors: 5/11
adequate: 2 low, 1 moderate, 0 high => risk=low, errors: 1/3
total errors: 6/14

4. Attribute *income*
0-15K: 0 low, 0 moderate, <u>4 high</u> => risk=high, errors: 0/4
15-35K: 0 low, <u>2 moderate</u>, 2 high => risk=moderate, errors: 2/4
over 35K: <u>5 low</u>, 1 moderate, 0 high => risk=low, errors: 1/6
total errors: 3/14

The rule based on the attribute *income* has the minimum number of errors =>1R produces the following rule:

if income=0-15K then risk=high
else if income=15-35K then risk=moderate
else if income=over 35K then risk=low

The new example has *income=15*-35K and will be classified as *risk=moderate*.

Question 7. (Type 2 – problem solving)

Use the k-means algorithm to cluster the following one dimensional examples into 2 clusters: 2, 5, 10, 12, 3, 20, 31, 11, 24. Suppose that the initial seeds are 2 and 5. The convergence criterion is met when either there is no change between the clusters in two successive epochs or when the number of epochs has reached 5.

Show the final clusters. How many epochs were needed for convergence? There is no need to show your calculations.

Solution:
For simplicity let's sort the data first:
2 3 5 10 11 12 20 24 31
K1={2}, K2={5}

At the end of epoch 1 the clusters are:
K1={2, 3}, mean_K1=2.5
K2={5, 10, 11, 12, 20, 24, 31}, mean_K2=16.1
Stopping criterion not met

At the end of epoch 2 the clusters are:
K1={2, 3, 5}, mean_K1=3.3
K2={10, 11, 12, 20, 24, 31}, mean_K2=18

Stopping criterion not met

At the end of epoch 3 the clusters are:
K1={2, 3, 5, 10}, mean_K1=5
K2={11, 12, 20, 24, 31}, mean_K2=19.6
Stopping criterion not met

At the end of epoch 4 the clusters are:
K1={2, 3, 5, 10, 11,12}, mean_K1=7.2
K2={20, 24, 31}, mean_K2=25
Stopping criterion not met

At the end of epoch 5 the clusters are:
K1={2, 3, 5, 10, 11,12}, mean_K1=7.2
K2={20, 24, 31}, mean_K2=25
Stopping criterion is met – no change in clusters

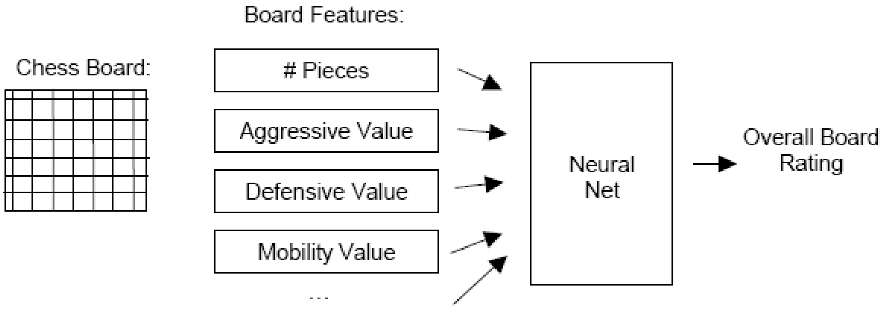
Note: The question asks you only to show the final clusters and number of epochs, so the answer is:

Clusters: K1={2, 3, 5, 10, 11,12}, K2={20, 24, 31}
5 epochs were needed.

Question 8. (Type 1 – short answers)

You task is to develop a computer program to rate chess board positions. You got an expert chess player to rate 100 different chessboards and then use this data to train a backpropagation neural network, using board features as the ones shown in the figure below.

Select the correct answer ("Yes" or "No") in the questions below. Select "Yes" for all issues that could, in principle, limit your ability to develop the best possible chess program using this method. Select "No" for all issues that could not. Briefly explain your answer.



a) The backpropagation network may be susceptible to overfitting, since you tested its performance on the training data instead of using cross validation.

Yes No

Explanation:

b) The backpropagation neural network can only distinguish between boards that are completely good or completely bad.

Yes No

Explanation:

c) The backpropagation network will converge to the global minimum.

Yes No

Explanation:

d) You should have used higher learning rate and momentum to guarantee convergence to the global minimum.

Yes No

Explanation:

e) The topology of your neural net might not be adequate to capture the expertise of the human expert.

Yes No

Explanation:

Answers:

Note: This is not a multiple-choice question. You need to provide an explanation, otherwise you will receive 0 marks.

a) Yes. Testing the performance on the training data is an over-optimistic measure and it does not guard against overfitting. Cross-validation is a more reliable procedure.

b) No. Backpropagation neural networks can be applied for both classification and regression tasks.

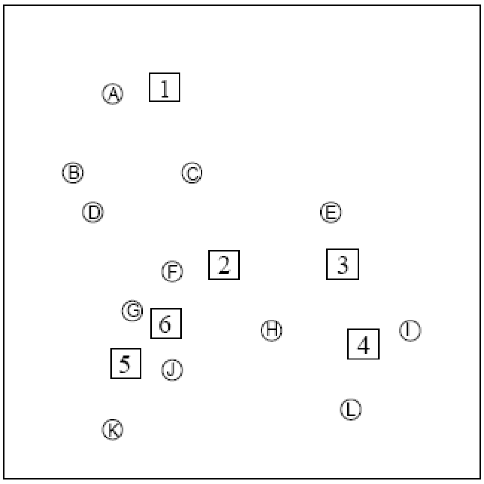
c) No. The backprogarion algorithm implements gradient descent and is not guaranteed to converge to the global minimum – it finds the closest local minimum.

d) No. The use of momentum reduces the oscillations when using a higher learning rate but it doesn't guarantee convergence to the global minimum.

e) Yes. Too few neurons – underfitting; too many – overfitting.

Question 9. (Type 2 – problem solving)

In the figure below, the circles are training examples and the squares are test examples, i.e. we are using the circles to predict the squares. Two algorithms are used: 1-Nearest Neighbour and 3-Nearest Neighbour.



We are given the following results about the squares:

Square	Using 1-Nearest Neighbors	Using 3-Nearest Neighbors
1	-	+
2	-	
3		+
4	+	-
5		-

What will be the class of the following examples? Write +, - or U for cannot be determined.

1) Circle L:
2) Circle I:
3) Circle H:
4) Circle E:
5) Circle K:
6) Circle C:
7) Square 6 using 1-Nearest Neighbour:
8) Square 6 using 3-Nearest Neighbour:
9) Square 3 using 1-Nearest Neighbour?
10) Square 5 using 1-Nearest Neighbour?

Answer:

Circle L: -
Circle I: +
Circle H: -
Circle E: +

Circle K: U
Circle C: +
Square 6 using 1-Nearest Neighbour: U
Square 6 using 3-Nearest Neighbour: -
Square 3 using 1-Nearest Neighbour: +
Square 5 using 1-Nearest Neighbour: U