UNIVERSITY OF OSLO

Faculty of Mathematics and Natural Sciences

IN3050/4050 — Introduction to Artificial Intelligence and Machine Learning
Exam Spring 2020
Exam content: Around 2 working days
Duration: From May 25 at 2:30 PM to June 2 at 2:30 PM
Permitted materials: All

- **General information about exams the spring 2020 can be found here.**
- **You should deliver your exam answers as a single PDF file through Inspera. Info on how to deliver files in Inspera can be found here.**
- **If you become ill and cannot deliver your written exam within the deadline, see here for procedures. There are given no postponements. There will be arranged a resit exam in August for they who get ill.**
- **You are free to make plots/figures in any program you want, and add them to your delivered PDF. That includes using drawing tools on the computer, and drawing figures by hand and taking a picture of them. Some tips are available here.**
- **Your delivery should be anonymous. Do not write your name.**
- **If you have questions related to this exam, please submit them here.**
- **Please check the "Messages" on the Course Website daily. We will post any clarifications around the content of this exam there.**
- **You may answer in English or Norwegian.**
- **You are not required to write any code for any of these tasks – all calculations may be done by hand, and we do want you to show each step of your calculation.**
- **The examination answers must be the result of the student's own efforts. It is okay to discuss theory and assignment text with others. It is also okay to get hints on how a task can be solved, but this should be used as a basis for your own answer and not be copied unchanged. Sharing code or (parts of) the solution of a task is not allowed. If you include text, program code, illustrations or other items from the internet or elsewhere, you must clearly mark it and indicate the sources – however, in an independent assignment this is something that rarely happens.**
- **The tasks of the exam are given points summing to 100, giving you an indication of how each task is weighted and roughly how much time you should spend on it.**

1) Search/Optimization (7p)

a) Draw a 1-dimensional search landscape (a continuous landscape f(x) over a single variable x). (1p)

Use it to explain the concepts (1p for each)

b) Local optima
c) The global optimum
d) Exploration
e) Exploitation

f) Is a pure-exploitation algorithm likely to find a local optimum? How about the global optimum? (2p)

2) Evolutionary Algorithm: The Social Distancing Game (14p)

The World Health Organization (WHO) have approached you with a task to help people maintain a safe distance from each other during virus outbreaks. As a simplifying assumption, assume we are dealing with optimizing the intra-person distance in a square room (NxN meters) where each 1x1 meter cell can hold one person at most (the WHO has stressed that any solution with more than 1 person in a cell is *invalid* and they don't want the EA to even consider such solutions). The room should contain exactly N people. Below is an example of a 4x4 meter room holding 4 people (indicated by X's).

X			X
	X		
		X	

a) Design an Evolutionary Algorithm capable of optimizing the social distancing in square rooms of NxN meters, with the objective of maximizing the minimal distance between any two persons in the room. Decide on and briefly justify your choice of the parameters below. (Be brief, but specific. The full answer to this should not exceed one A4 page – up to 2-3 lines should be more than enough to describe each point.):

-Genotypes. Show an example-genotype corresponding to the figure above.
-Phenotypes
-Fitness Function (write out the full calculation here, in mathematical formula or in code)
-Mutation Operator
-Crossover Operator
-Initialization Criterion
-Termination Criterion
-Selection Operator(s)

(0.75p for each)

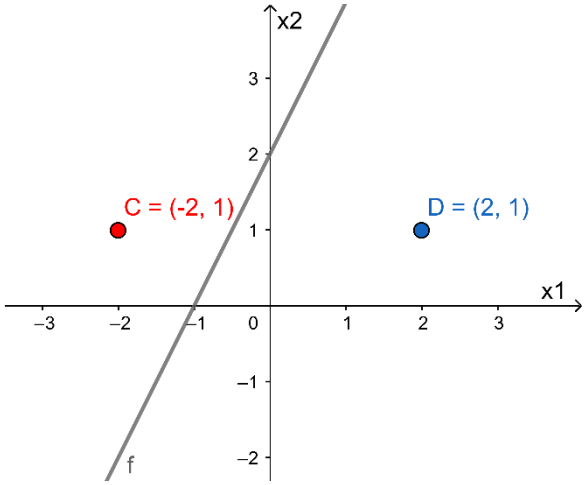
b) Are there any issues you have to pay special attention to in order to ensure your mutation, crossover and initialization does not produce invalid solutions? (2p)

c) You notice the EA suffers from premature convergence. Explain why fitness sharing can help you avoid premature convergence and describe how you could implement fitness sharing for this problem. (3p)

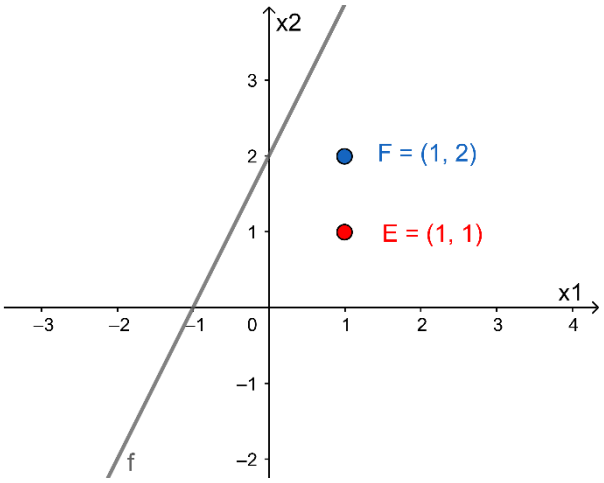
d) You're wondering if hybridizing the search can help you reach well-performing solutions faster.
   1) Suggest a way to hybridize this EA.
   2) Do you see any conflict between the goal of avoiding premature convergence and hybridizing the EA?
   3) Referring to the concepts of exploration and exploitation, how do the fitness sharing and hybridization affect the EA search?
   (3p total)

3) Classification (10 points)

We are considering a classification problem with two classes: The positive class, which we represent with the numerical value 1, and the negative class, 0. There are two features, x1 and x2, both are real numbers. We have trained a perceptron classifier on the training data. This has resulted in the decision boundary shown as line *f* in the figure. Points to the right of the line, e.g., the blue D = (2,1), are classified as 1, and points to the left of the line, like the red C = (-2, 1) are classified as 0.



a) Make an expression for how the classifier makes its decisions. Explain how you derived the expression. (Hint: Do not forget the bias). Show how this expression is used to classify C and D. (3p)



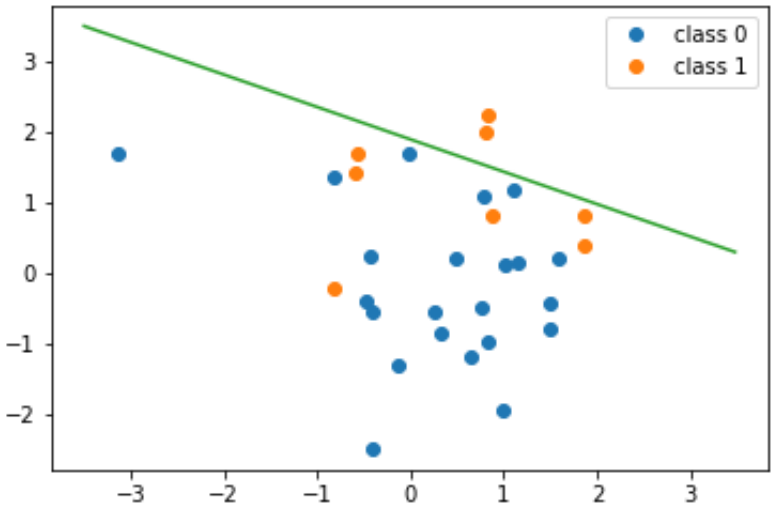
b) We pause during training, and we consider two observations from the training data: E = (1,1) which is wrongly classified, and, F = (1,2) which is correctly classified. Show how the perceptron training algorithm will update the weights when considering point E. Then show how it will update the weights when considering F after it has considered E. Make the necessary assumptions and state them clearly. (2p)

c) In a task like this, there are some parameters we have to set manually. One of them is the learning rate. There is no fixed value of the learning rate which fits all problems. Discuss with respect to the perceptron algorithm why we should make the learning rate not (too) big and not (too) small. (3p)

d) Suppose that you instead train a classifier using gradient descent for linear regression. Discuss why we should not make the learning rate (too) big nor (too) small in this case. (2p)

4) Evaluation (8 points)

We have trained a classifier and we are now ready to evaluate it on a test set of size 30. The result is as shown in the figure, where the green line indicates the decision boundary.



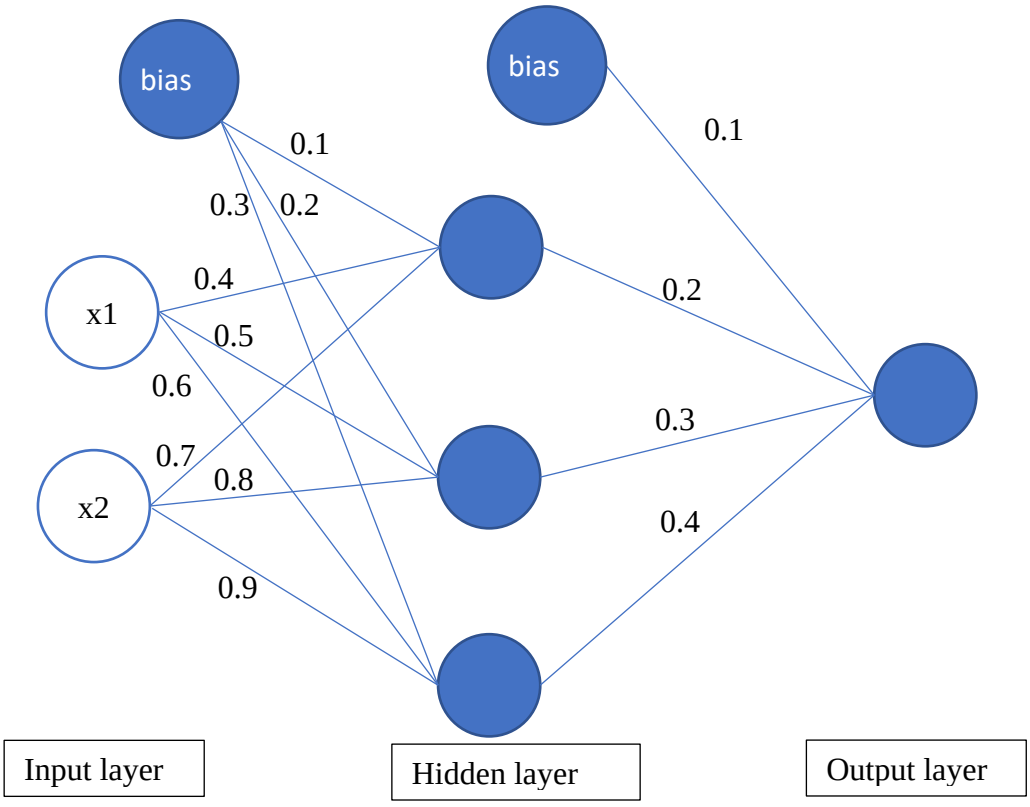
a) What is the accuracy of this classifier? Show how you found it. (2p)

b) How are the evaluation metrics precision and recall defined? What is the precision and recall of this classifier for each of the two classes? Show how you found them. (3p)

c) Formulate in your own words, why and under which conditions, we should use precision and recall, and not only rely on accuracy. Illustrate with examples of real-life problems. (3p)

5) Neural Networks (16 points)

We are using a feed forward neural network for solving a regression problem. The observations has m=2 features. There is one hidden-layer with n=3 nodes and a single output node. We assume that the logistic function is used as the activation function on the hidden layer. There is **no activation function on the output node** (since this is a regression problem.) At a point of time during training, the weights are as in the following figure.



a) Forwards step: Consider the observation **x** = (1, 2) What is the output value of the network for this observation? (5 p)

b) Backwards step: We assume we are training the network with stochastic gradient descent and will update the weights after each observation. The correct output for **x** in the training data is t = 10. Show how the weights are updated for this observation. Assume a learning rate of 0.1. Make additional assumptions when needed and explain them. (11 p)

6) Multi-class Classification (8 points)

a) What is meant by multi-class classification? (2p)

b) The goal is to classify pictures of fruit, where each picture contains one and only one kind of fruit. Say the classes are: apple, banana, grapes, orange, pear, plum. To prepare for machine learning, we have to represent these labels as numerical values. The first idea that comes to mind is to represent each class with an integer, say apple:0, banana:1, grapes:2, orange:3, pear:4, plum:5. Data sets are often stored and presented this way. Why isn't it a good idea to use these numbers directly as targets in an ML system, say a feed forward neural network? (1p)

c) A better proposal is to use "one-hot encoding", also called "one-out-of-n" encoding. Explain how that works and propose and encoding for the 6 classes. (2p)

d) One possible approach to multi-class classification is called "one vs. rest". Explain how it works. (3p)

7) Data Scaling (8 points)

You want to classify patients with respect to the probability of developing coronary diseases. You have a training material with data from many patients where several risk factors are measured, including the blood concentration of

- Cholesterol
- Triglyceride
- Natrium

All of them are measured in mmol/liter.

You consider all your data and calculate the min, max, mean and standard variation for each of the three and for the values considered together, and get the following table (the numbers are realistic, but constructed for this exercise):

	Cholesterol	**Triglyceride**	**Natrium**	**Collected**
Min	2.5	0.2	120	0.2
Max	10.5	3.0	150	150
Mean	6	1.3	140	49.1
Std. dev	2	0.7	5	48

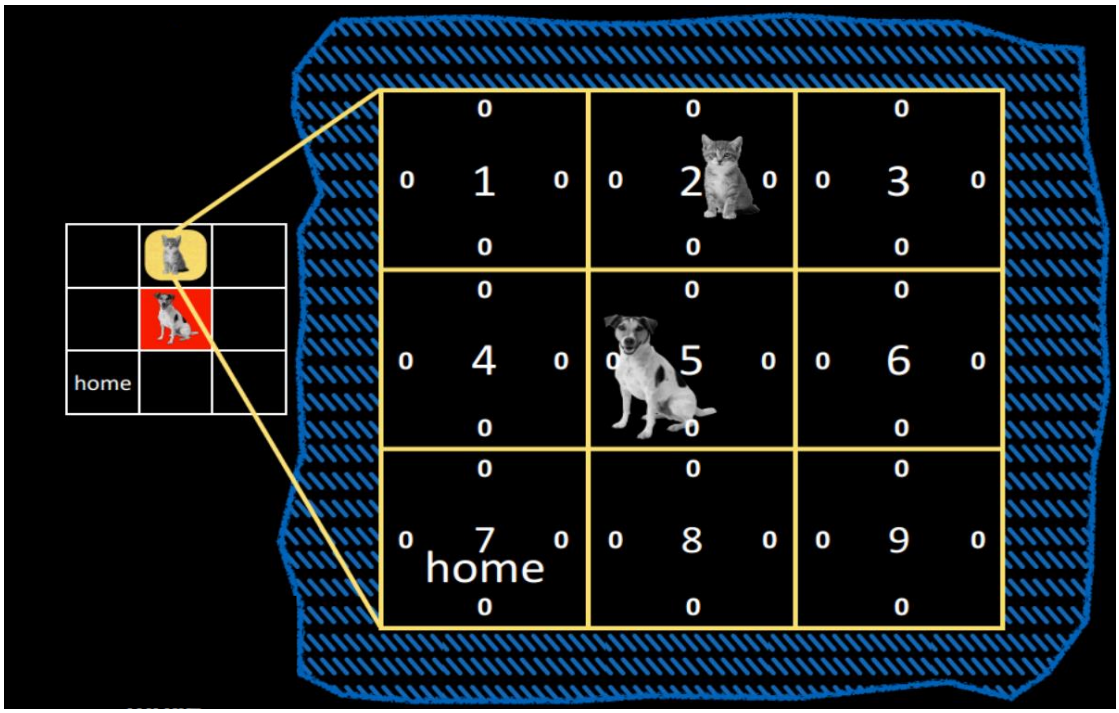
a) You have been advised to scale the data, and you choose to use min-max scaler. Explain the min-max scaler and show the result of scaling the following two observations. (2p)

A: (4.0, 2.0, 140)
B: (3.0, 0.1, 115)

b) You also want to see whether you get similar results by using the normal scaler (also called standard scaler). How is this different from the min-max scaler and what is the result of using this scaler to the two points? (2p)

c) Discuss the effect of scaling or not scaling your data with respect to the following three types of learners: *k* nearest neighbors (*k*NN), logistic regression, feed-forward neural network (MLP). (4p)

8) Q-learning (23p)



Remember this example from the lecture slides on Reinforcement Learning (Lecture Week 12, Part 2)? The example describes a cat trying to make its way home without bumping into a dog, which we formulate as a reinforcement learning problem.

The squares are the possible locations the cat can visit (**states**), numbered 1 to 9. The numbers in the edges of the squares are q-values, Q(s,a) describing the value of performing each possible **action** (move left, right, down, up) in that state. They are initialized to 0, and by using q-learning we want to update them to more accurately reflect the value of taking each action in each state.

If the cat moves outside the area defined by the 9 squares (e.g. by choosing the "up" action in State 2), it "falls off the table", and gets a negative reward, before being moved back to the state he came from.

The reward structure of the problem is as follows – **note that it is different from the reward structure in the original example from class**:

- Each movement by the cat gives a reward of -0.5 to encourage the cat to be efficient. Note that this reward is not applied when any of the other reward conditions below are present.
- Unlike in the lecture, this cat likes dogs, so it gets a reward of +5 for moving to the state of the dog (state 5).
- The cat gets a reward of -1 for "falling off the table".
- The cat gets a reward of +5 for moving to the "home" state (state 7).

We will use a discount factor of 0.9 and a learning rate of 0.2 (note: This is different from the example in the lecture).

a) Fill in the q-values of all states after the cat has performed 8 actions and updated the values with q-learning. The cat starts in state 2, and performs the following actions, receiving rewards and moving to new states as defined above. Action sequence: [Down, Right, Up, Right, Left, Down, Right, Down]. A reminder of the q-value calculation:

$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\mu}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$

Show the intermediate calculations. That is, if you update a given q-value twice, show the value after each update. (5p)

b) The second time the cat performs the "Down" action in state 2, the Q-value changes – even though the cat ends up in the same state as the previous time, and it gets the same reward. Why does it change? What would happen if the cat performed the "Down" action in state 2 very many times? Would the Q-value converge towards a specific value – and if so, which value? (you don't need to prove this mathematically, but explain the intuitions with your own words)
(3p)

c) With the q-values filled in after completing task a, what is the likelihood of choosing the action "down" in State 2, given you are using the following policies: 1) A greedy policy? 2) An epsilon-greedy policy with epsilon equal to 0.9? 3) A soft-max policy with a temperature of 1? Show your calculations and/or justify your answers.

Rank the policies with regards to how much they explore, and justify your ranking.
(4p)

d) What is the role of the discount factor? Why do we need to have one? What would happen to your calculated q-values if we reduced the discount factor to 0.1? What assumption do we make when we use a discount factor of 1?
(4p)

e) Assume we reset all q-values to 0. Perform the same actions, but update q-values according to the SARSA algorithm. You can assume that the actions in the sequence were actually the actions decided by the cat's policy. Since SARSA requires to know the consecutive action for updating q-values, you should perform only 7 updates of q-values this time, that is, do not try to update the q-value of the final "Down" action. Which q-values, will be different from the result in a?

For the values that are different, does off-policy or on-policy learning give the highest q-value? Referring to the assumptions made by q-learning on future rewards: why does on-policy learning and off-policy learning give different results here?
(5p)

f) We have made a little mistake in our reward function. Can you identify a form of "reward hacking" the cat can do, where it displays unwanted behaviour but still receives a high reward? (it is NOT visiting the dog – this cat really does like dogs)
(2p)

9) Particle Swarm Optimization (6p)

a) Explain how particles' position and velocity are updated in each iteration with reference to the formulas below. (2p)

$$x_{i,d}(it+1) = x_{i,d}(it) + v_{i,d}(it+1) \quad (1)$$

$$\begin{aligned} v_{i,d}(it+1) &= v_{i,d}(it) \\ &+ C_1 * Rnd(0,1) * [pb_{i,d}(it) - x_{i,d}(it)] \\ &+ C_2 * Rnd(0,1) * [gb_d(it) - x_{i,d}(it)] \end{aligned} \quad (2)$$

b) What happens if the information of the global best solution is not used in PSO? (2p)

c) PSO and evolutionary algorithms (EAs) are population-based search methods that have quite a lot of things in common. 1) In an EA, what concept corresponds to the particles in PSO? 2) In an EA, what concept corresponds to the positional parameters in PSO? 3) And finally, in an EA, what concept corresponds to/is related to the position update rule in PSO? Briefly justify your answers and any assumptions. (2p)