

**i Front Page IN3050**

# **IN3050 - Introduction to Artificial Intelligence and Machine Learning**

**Spring 2023**

**Tuesday, May 30**

**09:00 AM - 01:00 PM (4 hours)**

All questions should be answered!

Each question is assigned a weight which is indicated.

The maximum number of points for the whole set is 100 points.

Permitted materials: None

An on-screen calculator is available.

You may answer in English, Norwegian, Danish or Swedish.

---

## **1 Types of machine learning**

In this course, we have considered three main types of machine learning. What are they? For each of the three, describe shortly (1-2 sentences) a task where one could apply this type of machine learning.

**Fill in your answer here**

---

Maximum marks: 6

### **Solution proposals**

- **Supervised learning:** Can be used for classifying e-mails into spam or no-spam.
- **Unsupervised learning:** Can be used for clustering readers of news stories on the background of the stories they read.
- **Reinforcement learning:** Can be used for learning to play games e.g., checkers or chess.

Guide: Up to 2 points per answer.

## 2 Travelling Salesperson - Search

We are trying to solve a Travelling Salesperson Problem. Write code(\*) for solving this problem with

- a) A hillclimbing algorithm (6 points)
- b) Simulated annealing (9 points)

Each algorithm should run for 200 iterations, that is, 200 total evaluations of TSP solutions. You may decide the annealing schedule for simulated annealing yourself.

Assume we have the following methods available, and use them in your code:

GENERATE\_RANDOM\_NEIGHBOR(some\_solution): Returns a "neighbor" of some\_solution, that is, a solution that is identical to some\_solution, but with exactly two cities swapped. For instance, for the TSP solution [1, 2, 3, 4] it could return [1, 2, 4, 3].

GENERATE\_RANDOM\_SOLUTION(): Generates a random valid (each city visited exactly once) solution to our TSP-problem.

EVALUATE\_DISTANCE(some\_solution): Returns the distance travelled for the salesperson given the plan some\_solution.

SHOULD\_I\_EXPLORE(better\_solution, worse\_solution, current\_temperature): Returns True or False. Implements the simulated annealing logic where a worse solution is sometimes selected, depending on the temperature. If True, the decision is to "explore", meaning we accept worse\_solution. If False, we instead keep better\_solution.

(\*) We will not be strict about the formatting/syntax of your code - focus on making it understandable. If you want, you can write something similar to Python code. To get full score, you have to *keep track of the variables and show how the result is calculated from the initial values*.

Example of how the code could look for an algorithm looking for the smallest number in a list of positive integers (notice that the code keeps track of the relevant variables and shows all steps of the calculation):

```
#Assuming the input list is given in the variable a_list_of_numbers
```

```
smallest_num_so_far = a_list_of_numbers[0]
```

```
for(number in a_list_of_numbers):
```

```
    if(number<smallest_num_so_far):
```

```
        smallest_num_so_far = number
```

```
#smallest_num_so_far now contains the smallest number.
```

**Fill in your answer here**

---

Maximum marks: 15

## Suggested Solution

a)

Possible code for Hillclimbing:

```
best_solution = GENERATE_RANDOM_SOLUTION()
best_distance = EVALUATE_DISTANCE(best_solution)
for(num_iterations in range(0,200)):
    new_solution=GENERATE_RANDOM_NEIGHBOR(best_solution)
    new_distance=EVALUATE_DISTANCE(new_solution)
    if(new_distance < best_distance):
        best_solution = new_solution
        best_distance = new_distance
#After the loop finalizes, the result of hillclimbing is in best_solution.
```

Pseudocode may differ from the suggestion above, points should be awarded for how much of the algorithm the student has understood. Approximate scores:

2 points for initializing a TSP-solution and its score

2 points for looping through and generating neighbors. Note: The solution above always generates just 1 neighbor, and then continues iterating. But it would also be ok to generate more neighbors and choose the best one of them.

2 points for correctly replacing neighbors in each iteration (and not replacing them if the neighbor is worse)

b)

Possible code for Simulated annealing:

```
current_temperature = 20
best_solution = GENERATE_RANDOM_SOLUTION()
best_distance = EVALUATE_DISTANCE(best_solution)
for(num_iterations in range(0,200)):
    new_solution=GENERATE_RANDOM_NEIGHBOR(best_solution)
    new_distance=EVALUATE_DISTANCE(new_solution)
    if(new_distance < best_distance):
        best_solution = new_solution
        best_distance = new_distance
    else if(SHOULD_I_EXPLORE(best_solution, new_solution, current_temperature)):
        # Note: SHOULD_I_EXPLORE is only checked if the old solution was best, since we always want to
        # keep new, better solutions.
        best_solution = new_solution
        best_distance = new_distance
    current_temperature-=0.1
#After the loop finalizes, the result of simulated annealing is in best_solution.
```

Pseudocode may differ from the suggestion above, points should be awarded for how much of the algorithm the student has understood. Approximate scores:

2 points for initializing a TSP-solution, its score and the temperature

2 points for looping through and generating neighbors

3 points for correctly replacing neighbors in each iteration - where 1) the new solution is always kept if it is better (if `new_distance < best_distance`), and 2) otherwise, the probability of choosing worse solutions should have some randomness, and be influenced by the temperature

(`SHOULD_I_EXPLORE(best_solution, new_solution, current_temperature)`). Partial points may be awarded if only some of this is present.

- ➔ Note especially here that if we do not check first that `new_distance < best_distance`, we do not know which solution is best, and can therefore not use the `SHOULD_I_EXPLORE` method correctly. It is also important that, if `new_distance < best_distance` returns `True`, we should not call `SHOULD_I_EXPLORE` (or we can ignore its output): If the new solution is better we always want to keep it.

2 points for reducing the temperature sensibly (e.g. a little bit each iteration or with larger jumps but not as frequently).

### 3) EA Representations – Example (3050) (4p)

#### **3 Evolutionary Algorithms - Representations Example**

Give an example of an optimization problem we could solve with an Evolutionary Algorithm, a relevant representation for the solutions to that problem, and a suitable variation operator.

**Fill in your answer here**

---

Maximum marks: 4

#### Suggested Solution

Many possible correct solutions here. 1 point for a relevant problem, 1.5 points for a relevant representation and 1.5 points for a variation operator that works for that representation.

Possible answers:

- Travelling Salesperson Problem, solutions represented as permutations, Partially-matched Crossover (PMX)
- Optimizing the design of a robot, solutions represented as lists of real numbers, uniform mutation.

It is not necessary to give details of the variation operators or representations for a full score here. The name is enough if it is obvious how they are related to the proposed problem.

### 3) EA Representations – efficiency (4050) (4p)

#### 3 EA Representations - efficiency

How can the choice of representation type for solutions make the search in an Evolutionary Algorithm more or less efficient? Give an example, with a specific optimization problem and two different solution representations where one is likely to give a more efficient search than the other.

**Fill in your answer here**

---

Maximum marks: 4

#### Suggested Solution

The representation allows us to encode some knowledge about the problem we are trying to solve and how to efficiently move from one solution to the next. We can also be able to choose a representation that only allows certain kinds of solution, allowing us to avoid searching for those solutions that are not admitted by the representation. (2p)

Example (2p):

Any representation that is likely to be more efficient than another could award points here. The most straightforward example is to use a permutation-type problem, like a travelling salesperson problem. A permutation representation here allows us to avoid searching for solutions that are invalid (because they are not permutations) thus saving time. A less efficient representation would be for instance an integer-style representation, that could represent solutions such as [1 1 2 1] which are not useful for the TSP search.

### 4a) Increasing Exploration in RL (IN3050) (3p)

#### 4(a) Increasing exploration in Reinforcement Learning

Describe a way to increase the amount of exploration in a Reinforcement Learning algorithm.

**Fill in your answer here**

---

Maximum marks: 3

#### Suggested Solution

The main way we have covered of doing this in the lecture is by changing the policy. That could for instance be to go from a greedy policy to epsilon-greedy, or if already using epsilon greedy to increase the chance of a random choice (by increasing the value of the epsilon-parameter). Other sensible answers could also be valid. (3 points)

#### 4a) Adaptive Exploration – RL (4050) (3p)

##### 4(a) **Adaptive exploration - RL**

Describe a way to adaptively adjust the amount of exploration through the search/learning process in Reinforcement Learning. Should exploration increase or decrease as the search/learning progresses?

**Fill in your answer here**

---

Maximum marks: 3

##### Suggested Solution

The main way we have covered of doing this in the lecture is by changing the policy. Adaptively changing its exploration would most likely be done by reducing the value of epsilon in an epsilon greedy policy as search progresses. Other sensible answers could also be valid. (2 points).

Exploration should decrease as search progresses (1 point)

#### 4b) Increasing Exploration in an EA (IN3050) (3p)

##### 4(b) **Increasing Exploration in EAs**

Describe a way to increase the amount of exploration in an Evolutionary Algorithm

**Fill in your answer here**

---

Maximum marks: 3

##### Suggested Solution

There are many ways to increase the amount of exploration in an EA, e.g. by increasing the probability or impact of mutations/crossover, or decreasing the selecting pressure by having more randomness in parent selection or survivor selection. Other solutions that lead to increased exploration could also be accepted (3 points).

#### 4b) Adaptive Exploration – EA (4050) (3p)

##### 4(b) **Adaptive exploration - EA**

Describe a way to adaptively adjust the amount of exploration through the search process in an evolutionary algorithm. Should exploration increase or decrease as the search progresses?

**Fill in your answer here**

---

Maximum marks: 3

### Suggested Solution

There are many ways to adaptively adjust exploration in an EA, e.g. by reducing the probability or impact of mutations/crossover as search progresses, or increasing the selecting pressure by having less randomness in parent selection or survivor selection as search progresses. (2 points).

Exploration should decrease as search progresses (1 point)

#### 5(a) Training and test set

In supervised machine learning experiments, we consider two types of data, training data and test data. What are the respective functions of the two sets?

Fill in your answer here

---

Maximum marks: 3

#### 5(b) Reasons

What are the reasons for introducing two different sets, and why do we need to keep them separated?

Fill in your answer here

---

Maximum marks: 3

#### 5(c) Development test set

We sometimes also consider a third type of data, called a *validation set*, or a *development test set*, or more simply a *development set*. What is the role of this set? Why do we need both the development set and the test set?

Fill in your answer here

---

Maximum marks: 3

#### 5(d) Cross-validation

An alternative is to use cross-validation, e.g., 10-fold cross-validation. Describe shortly how this works. Why should one prefer to use cross-validation?

Fill in your answer here

---

Maximum marks: 3

### Solution proposals

- a) The goal of supervised ML is to train an algorithm which can be applied to new and unseen data, e.g., to classify new and incoming mail as spam or not. The training data is used for training an algorithm. The test data is used for measuring how well the algorithm will perform on new and unseen data.
- b) There is a danger the algorithm has adapted too well to the training data, and will perform better on the training data than on new and unseen data. Therefore, it is important that the system is tested on test data not seen during training.
- c) When we try to train the best algorithm for a certain task, we may train various learning algorithms and various hyperparameter settings on the training data to see which works best. We need a test set for repeated testing of the various algorithms and hyperparameters. We use the development test set for this. What turns out as the best learning algorithm and settings can depend on properties of the particular development test set. Hence, we need a different, clean set for final testing.
- d) For n-fold cross-validation, we can split the training set into n equally sized bins,  $S_1, S_2, \dots, S_n$ . We can run n many experiments. In experiment k, we use  $S_k$  as test set and the rest as training set. We can take the averages of the n many experiments as evaluation measures. The n algorithms will be partly different since they are trained on slightly different training sets. However, most of the training data are the same between two experiments; hence, the average gives a rough measure of what we may achieve by training on the whole training set. The main advantage of using cross-validation is we get a larger test set, the whole training set. Hence, the numbers are more reliable when we do not have many data.

### Exercise 6 (IN3050 only) (6 points)

#### 6 Loss

We are considering a regression problem, where the goal is to predict a value  $y$  from a simple observation  $x$ . We have three observations

- $x_1 = -1, y_1 = 2$
- $x_2 = 0, y_2 = 0$
- $x_3 = 1, y_3 = 0$

(This is highly simplified to make it manageable by hand with limited time.)

We are using linear regression for the prediction and we train it by gradient descent. So far, the trained model is  $y = x + 1$ . What is the mean square error between the observations and the model?

**Fill in your answer here**

---

Maximum marks: 5

### Solution proposals

- a) We use  $z$ -s for the predicted values and keep the  $y$ s for the target values.

Predicted values:

- $z_1 = x_1 + 1 = -1 + 1 = 0$
- $z_2 = x_2 + 1 = 0 + 1 = 1$
- $z_3 = x_3 + 1 = 1 + 1 = 2$

$$\text{MSE: } \frac{1}{3} \sum_{j=1}^3 (y_i - z_i)^2 = \frac{1}{3} ((2 - 0)^2 + (0 - 1)^2 + (0 - 2)^2) = 3$$

## 7(a) Perceptron

The perceptron was an early approach to machine learning. It was introduced and became popular around 1960. Describe the architecture of a perceptron and how it can be applied for classification. You do not have to describe how the perceptron is trained.

Fill in your answer here

---

Maximum marks: 6

### Solution proposals

- The perceptron is an algorithm for binary classification.
- The input should be a numerical vector of a certain length  $x = (x_1, x_2, \dots, x_m)$ .
- If the length is  $m$ , the perceptron contains  $m$  many weights,  $w_1, w_2, \dots, w_m$ , and a threshold  $w_0$ .
- If the weighted sum,  $s = \sum_{i=1}^m w_i * x_i$  is larger than the threshold  $w_0$ ,  $x$  is ascribed to the positive class. If  $s \leq w_0$ ,  $x$  is ascribed to the negative class.

This can also be described as adding one additional feature,  $x_0 = -1$  (or  $x_0 = 1$ ) to each input together with an additional weight  $w_0$  and consider whether  $\sum_{i=1}^m w_i * x_i > 0$  or not.

## 7(b) Neural nets

Give short answers to each of the following questions:

1. The perceptron has its limitations. When attention was directed towards these limitations towards the end of the 1960s, the interest for perceptrons diminished. What were these limitations?
2. Why did neural network again catch researchers' interest in the second half of the 1980s?
3. In spite of these discoveries, neural networks had to wait another, roughly, 25 years before they reached results superior to other approaches. What were the main factors that made neural networks succeed from about 2012?

Fill in your answer here

---

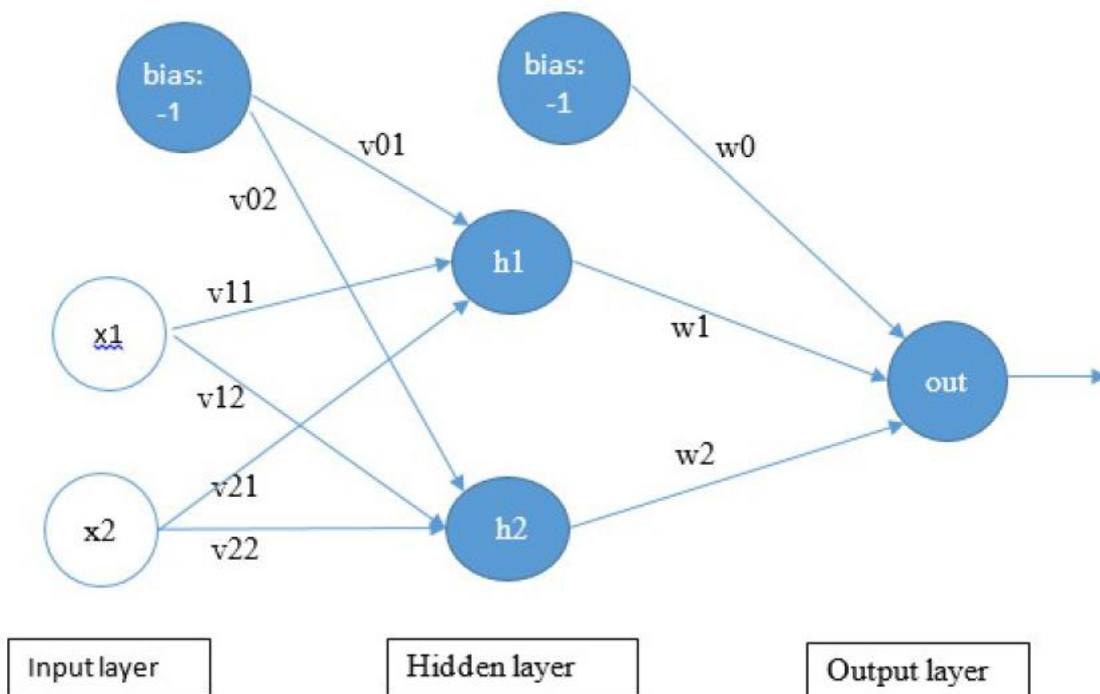
Maximum marks: 6

## Solution proposals

### Answers

1. A perceptron will always give a linear decision boundary. It cannot be used for, e.g., the logical function XOR. By adding hidden layers, one can model non-linear problems. However, by the end of the 1960s they did not know how to train networks with hidden layers.
2. In the mid 1980s, it was published a paper which showed how to train a neural network with hidden layers by gradient descent and backpropagation.
3. It was a combination of several factors
  - a. Faster hardware, in particular due to using GPUs for processing.
  - b. Larger amounts of training data, thanks to WWW.
  - c. Better algorithms using deep neural networks.

8



Consider the network in the figure. This is a network with one layer. There are two regular input nodes, one output node and two regular nodes in the hidden layer. In addition, there are added bias nodes in each layer, each with the value -1. The weights are indicated by symbols, e.g., the weight for the connection from input node 1 to hidden node  $h_2$  is  $v_{12}$ . Moreover, the logistic function (sigmoid) is used as activation in the hidden layer.

In the following, the goal is to describe this network by formulas using the symbols, and thereby describing the algorithms both for how the network is used for predicting and for how it is trained.

### 8(a) Logistic function

Give a general formula for the logistic function.

Fill in your answer here

---

Maximum marks: 3

Solution proposal

$$\text{logistic}(x) = 1/(1 + e^{*(-x)})$$

It is here also correct to answer  $\text{logistic}(x) = 1/(1 + e^{*(-b*x)})$ , where  $b$  is a constant, which is the form provided by Marsland, or the more general  $\text{logistic}(x) = 1/(1 + e^{*(-b*(x-x_0))})$ ,  $x_0$  also a constant. But the simpler form suffices.

### 8(b) Output from hidden layer

Give a formula for the output from the hidden node  $h_1$  given a general input  $(x_1, x_2)$ . You may in this point simply refer to the logistic function as  $f(x)=\text{logistic}(x)$ . This means that your answer here will not depend on whether your formula for the logistic function was correct or not.

Fill in your answer here

---

Maximum marks: 3

Solution proposal

The value into  $h_1$  is  $h_1 = ((-1) * v_{01}) + (x_1 * v_{11}) + (x_2 * v_{21})$

The value out of  $h_1$  is  $a_1 = \text{logistic}(h_1) = 1/(1 + e^{*(-h_1)})$

### 8(c) Output

The network is used for classification. We assume a logistic activation also in the output layer. Give a formula for the output value  $y$  from the *out* node for the input  $(x_1, x_2)$ .

You may use the output value  $a_1$  from question (b) above directly, and similarly for  $a_2$ . You do not have to copy in the solution from the previous questions.

Fill in your answer here

---

Maximum marks: 3

Solution proposal

Sufficient answer: The value into the out node is  $z = (-1) * w_0 + a_1 * w_1 + a_2 * w_2$ , and the out value is  $y = \text{logistic}(z) = 1/(1 + e^{*(-z)})$

(Not necessary to include for full score:)

$a_2$  can be expressed similarly to  $a_1$  as  $a_2 = \text{logistic}(h_2) = 1/(1 + e^{*(-h_2)})$ ,

where  $h_2 = ((-1) * v_{02}) + (x_1 * v_{12}) + (x_2 * v_{22})$ .

### 8(d) Update final layer

Assume that the target value for  $(x_1, x_2)$  is the positive class,  $t=1$ . We are training the network with stochastic gradient descent updating after each observation. Assume a learning rate of 0.1. Give a formula for the weight  $w_2$  from  $h_2$  to *out* after updating for  $(x_1, x_2)$ .

You may also here reuse the values you have already expressed.

**Fill in your answer here**

---

Maximum marks: 3

#### [Solution proposal](#)

The delta term at the out node is  $d_0 = y - t = y - 1$ .

Hence, the new value will be  $w_{2\text{new}} = w_2 - 0.1 * d_0 * a_2 = w_2 - 0.1 * (y - 1) * a_2$

Comment: We have assumed (binary) cross-entropy loss. If one instead uses MSE-loss, the answer for  $d_0$  is  $d_0 = (y - t)y(1-y) = (y - 1)y(1-y)$ . This will also give a full score here.

### 8(e) Update hidden layer

Give a formula for the new weight  $v_{12}$  from  $x_1$  to  $h_2$  after the same round of updating.

You may also here reuse the values you have already expressed.

**Fill in your answer here**

---

Maximum marks: 3

#### [Solution proposal](#)

To update  $v_{12}$ , we first have to find the delta term on  $h_2$ , call it  $dh_2$ . This must be done before we perform the update of  $w_2$  (point d above).

$dh_2 = a_2 * (1 - a_2) * w_2 * d_0$ , where  $a_2 * (1 - a_2)$  is the derivative of the activation function on  $h_2$ . Then  $v_{12\text{new}} = v_{12} - 0.1 * dh_2 * x_1$

IN4050-only

### 7(f) Loss function

Which loss function did you use, and what was the loss for  $(x_1, x_2)$ ,  $t=1$  before updating? Give both the name of loss function and, preferably, a formula for the loss before updating.

**Fill in your answer here**

---

Maximum marks: 3

Solution proposal

As loss function, we used (binary) cross-entropy loss,  $L(y, t) = t \log(y) + (1 - t) \log(1 - y)$   
(2 points for only name or only formula, 3 points for both)

Alternatively, one could use the mean square error loss. But then the delta term in (d) must be expressed  $d_0 = (y - t)y(1-y)$

IN4050-only

### 7(g) ReLU

Assume that you instead use ReLU (Rectified Linear Unit) as the activation function in the hidden layer. Give a formula for the output from the hidden node  $h_1$  given a general input  $(x_1, x_2)$  with ReLU as activation in the hidden layer.

**Fill in your answer here**

---

Maximum marks: 2

Solution proposal

$\text{ReLU}(h_1) = h_1$  if  $h_1 > 0$  and  $\text{ReLU}(h_1) = 0$  if  $h_1 < 0$ .

|   |   |   |   |    |   |
|---|---|---|---|----|---|
| 0 | 0 | 0 | 0 | r  | 0 |
| 0 | 0 | 0 | 0 | 10 |   |
| 0 | 0 | 1 | s | 10 | 0 |
| 0 | 0 | 1 |   | 0  |   |
| 0 | 0 | 0 |   | 0  |   |
| 0 | 0 | 0 |   | R  | 0 |
| 0 | 0 | 0 |   | 0  |   |

9(a) Reinforcement Learning - policy 1

Consider the Reinforcement Learning grid world on the left. Each square is a state, and numbers represent Q-values associated with moving to the neighbor state in the given direction. Capital "R" is a large reward, while lower-case "r" is a small reward. "S" is the start state. Note that the Q-values at the moment do not really reflect the location the reward in the environment.

Assuming the agent follows a greedy policy, and each episode lasts for 5 steps, will an agent ever find the large reward? Why/why not? To keep things simple, you can assume that the agent will not update its Q-values.

Fill in your answer here

Maximum marks: 5

### Suggested Solution

The agent will not find the large reward, because when it reaches the right center state, it will always go up due to following a greedy policy. A greedy policy means always choosing the action with the highest associated Q-value. (5p)

## 9(b) Reinforcement Learning - Policy 2

Suggest one other policy we could choose to help the agent find the largest reward. With reference to the example on the left, show how that policy leads the agent to the correct place.

**Fill in your answer here**

---

Maximum marks: 5

### Suggested Solution

This could be done with an epsilon-greedy policy or a softmax policy (mentioning one is enough). In both cases, one could choose parameters (the epsilon in the former case, and the  $\tau$  in the latter) that lets the agent select the greedy option most of the time, and a small percentage of the time instead explores the other states. (3p)

For the grid world in the figure, that would have the effect that the agent may first take a greedy choice and move right, then it may later randomly explore and visit the bottom state. (Note, this would probably not happen in the first episode, but after many episodes the agent will most likely be lucky to explore just at the right moment, and end up with the high reward.) (2p)

## 10(a) k-means clustering

Briefly describe the different steps in the K-means clustering algorithm.

**Fill in your answer here**

---

Maximum marks: 5

### Suggested Solution

There are 3 main steps:

First, randomly select  $k$  cluster centers

Then, repeat until convergence:

- 1) Assign each data point in our dataset to the nearest cluster center
- 2) Update the cluster centers to the mean of the samples in that cluster

## 10(b) Autoencoder

What is an autoencoder neural network? Name one possible usecase for such a network.  
**Fill in your answer here**

---

Maximum marks: 5

### Suggested Solution

An autoencoder is a neural network that learns to produce an output similar to the input (2.5p) We cover a few usecases in the course – mentioning one is enough: Learning to remove noise, learning to generate new data from the same distribution (e.g. by learning a low-dimensional representation of the data, and perturbing it slightly to get something new), learning to compress data. Other valid usecases can also be accepted. (2.5p)

## 11 AI Ethics - Biases

What do we mean with the problem of biases when applying Machine Learning algorithms? Give an example of an application where biases in the training data can be problematic.  
**Fill in your answer here**

---

Maximum marks: 5

### Suggested Solution

Note: The answers don't have to be as long as the suggestion below to get a full score. A short but correct explanation is perfectly fine.

The problem of biases in machine learning refers to the presence of unfair or discriminatory outcomes that can arise when training algorithms on biased data or when the algorithms themselves inadvertently learn and perpetuate biases present in the data. These biases can manifest in various forms, such as racial, gender, or socioeconomic biases, and can lead to unfair treatment or inaccurate predictions for certain groups. (2.5p)

An example of an application where biases in the training data can be problematic is in the field of hiring or recruitment. Let's consider an automated resume screening system that is trained on historical hiring data. If the historical data exhibits biases, such as favoring candidates from specific demographics or educational backgrounds due to systemic biases in the hiring process, the machine learning algorithm may inadvertently learn and reinforce these biases.

For instance, if the training data predominantly consists of resumes from male candidates who were historically favored in the hiring process, the algorithm may learn to associate male attributes with a higher likelihood of being selected. As a result, the algorithm might unfairly disadvantage female candidates in the screening process, perpetuating gender biases. (2.5p)

---