

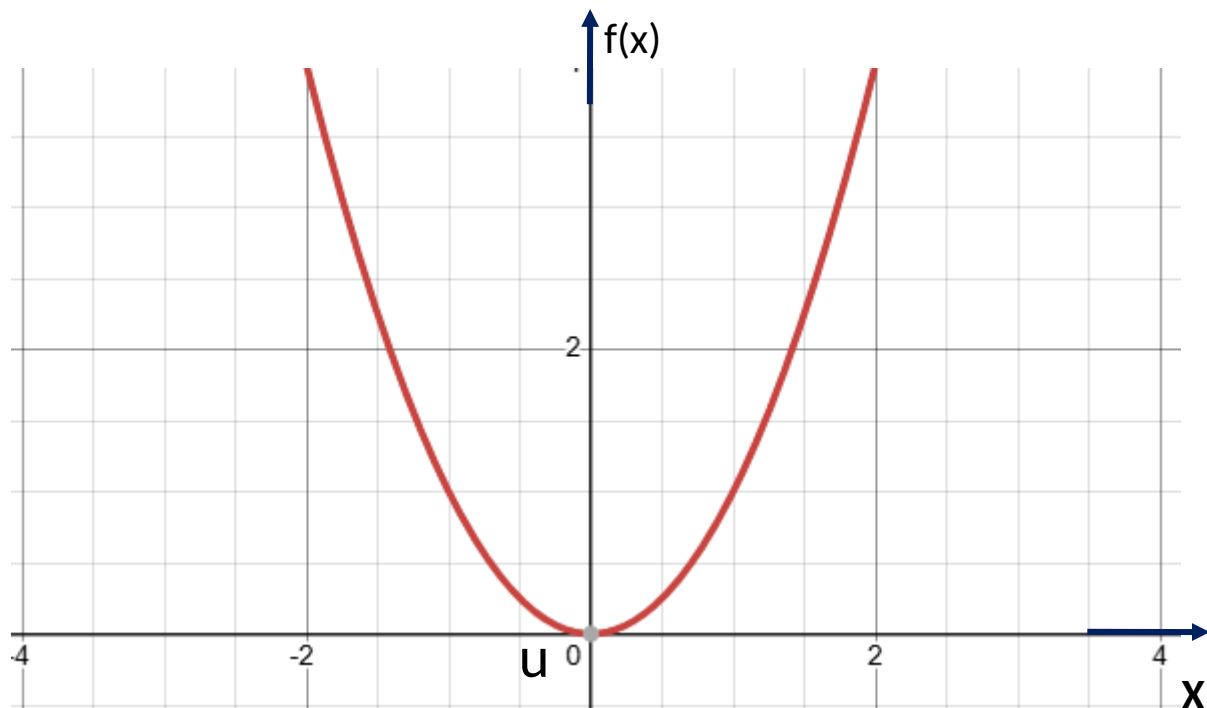
Exam IN3050/IN4050 2025 – Suggested Solutions

Search and Optimization (7p)

The figure below shows a 1-dimensional search landscape for a *minimization* problem. We are optimizing a variable x , and the quality of our solution varies as shown depending on the value of x . We start searching for a solution at the value $x=u$.

Different search methods have different ways of finding the next solution to evaluate. Let's call the next evaluated point in the search landscape v . Describe in words how v can be selected, given the search methods

- Gradient descent. Mention how v depends on the descent rate/ step size (called α in the course textbook by Marsland) (3.5p)
- Local search / hillclimbing (or, more accurately hill descending in our case). State any assumptions you make around how neighbor solutions are explored (what Marsland calls the move-set). (3.5p)



Suggested solution

- The next point (v) is determined by moving from (u) in the direction of the negative gradient of the function at (u). The amount of movement in the given direction is affected by the step size α .

Mathematically, this can be written as $v = u - \alpha * \nabla f(u)$. Either the mathematical or textual description is enough for a full score here.

- b) In local search methods such as hillclimbing, (v) is selected from the neighboring solutions around the current solution ($x = u$) based on the move-set, which defines how neighbors are explored. For instance, if we assume that the move-set contains the solutions that are 0.1 above or below the current solution, the explored solutions would be $v_1 = u + 0.1$ and $v_2 = u - 0.1$. Neighbors could also be chosen in different ways, the main point here is that there is no gradient to follow, so neighbors instead have to be chosen by systematically or randomly making a small change to the current solution. Reasonable descriptions of how to do so get 3.5 points.

Reinforcement Learning – IN3050 (5p)

Write down the formula for updating Q-values in Q-learning. Explain the meaning of each element in the formula. If you don't remember the exact variables used, explain verbally the different elements in the formula.

Suggested solution

The Q-learning formula is as follows:

$$Q(s, a) \leftarrow Q(s, a) + \mu(r + \gamma \max_{a'} Q(s', a') - Q(s, a))$$

Elements Explained:

- **$Q(s, a)$** : Current estimate of the return for taking action (a) in state (s).
- **μ** : Learning rate; determines how much the Q-value is updated.
- **r** : Immediate reward for action (a) in state (s).
- **γ** : Discount factor; balances immediate vs. future rewards.
- **$\max_{a'} Q(s', a')$** : Highest predicted Q-value for actions in the next state (s'), guiding the estimate for future rewards.

Together, these elements update ($Q(s, a)$) by incorporating new experiences to gradually improve policy decision-making.

Formula and explanation gives full score, either part alone gives half score.

Reinforcement Learning – IN4050 (5p)

- a) What is the role of the discount factor in Reinforcement Learning? (2.5p)
- b) Suggest a Reinforcement Learning scenario where it would be wise to have a relatively low discount factor. Briefly state the reasoning behind your answer. (2.5p)

Suggested Solutions

- a) The discount factor (γ) in reinforcement learning determines how much future rewards are valued compared to immediate ones. A factor close to 1 emphasizes long-term benefits, while a factor near 0 prioritizes immediate rewards.
- b) A situation where it would be wise to have a relatively low discount factor is a situation where the far future is very uncertain, and we should therefore not give it too much attention when selecting our actions. This could be the case e.g. when playing a board game (e.g. chess) against a very unpredictable opponent: We should focus on making smart decisions here and now, but we can't really tell how they will affect where the game stands many moves from now. Other sensible suggestions for scenarios where our actions' effect on the far future are uncertain should also get a full score here.

AI evaluation (6p)

In the history of AI, it has been common practice to evaluate new AI approaches and models on their performance in playing board games such as Chess or Go. Provide two arguments in favour of, and two arguments against this evaluation method.

Arguments in favour (1.5 points per bullet point below, max 3 points):

- Board games have clear rules and it is clear what is considered a successful outcome (winning the game), so evaluation is quite easy.
- Board games are a testbed where humans actually care about winning and need a high amount of (a certain type of) intelligence to win.
- It's a relatively equal playing field for humans and machines (don't need a body).
- In the recent past, game AI systems have come up with new strategies that human players didn't use. This suggests that AI systems show some genuine intelligent behaviour, and it therefore makes sense to include board games in AI testing.

Arguments against (1.5 points per bullet point below, max 3 points):

- Winning board games doesn't necessarily require intelligence - many games can be solved with high amount of computation + some heuristics.
- Focusing on games ignores many important aspects of human intelligence: social/emotional aspects, embodied aspects, sensing, ...
- The first aspect above could be linked to the Chinese room argument: if one has an exhaustive list of game states and their preferred next move, then all it takes to identify the current game state in the list, which isn't really "intelligent behaviour".

Supervised learning: features (6p)

- a) Explain what do the terms "feature" and "label" mean in the context of supervised machine learning.
- b) What feature types do you know?
- c) How they are related to the notion of "representation"?
- d) Come up with an example of a classification problem, pointing out what exactly serve as features and labels in this specific problem.

Suggested solution

a) Features are representations of some properties of the objects we would like to apply supervised learning algorithms to. They are the input of any classification system. We hope that these properties are informative enough to make it possible to predict the **label** of the object based on them. Label denotes which class the object belongs to. Labels are the output of a classification system. **(2 points)**

b) It is enough to mention **numerical** and **categorical** features. **(1 point)**

c) The students are expected to explain that numerical features represent object properties as numerical values (integers or floats), e.g., “4”, while categorical features represent properties as categories from a closed set, e.g., “red”. Depending on how we **represent** the objects (data instances), our supervised learning systems will behave differently. That’s why feature selection is so important. **(1 points)**

d) The students can come up with any sensible example. The one from the lectures is **spam detection**, where the objects are email messages, represented with numerical features (length in characters, number of line breaks, etc) and categorical features (occurrence of the word “buy”, etc). Labels in this problem are binary: “spam” or “not spam”. **(2 points)**

Supervised learning: batches (6p)

When training a model using gradient descent, one can use several different approaches: stochastic gradient descent (SGD), batch training and mini-batch training (among others). Describe these varieties, their advantages and disadvantages. Can some of them be considered a subset of others?

Suggested solution

All of these approaches use gradient descent, the only difference is how many data instances are considered to conduct one learning step (compute the loss and update the weights). In the “vanilla” **batch gradient descent**, one learning step processes **all** the instances in our training dataset. In the **mini-batch** setup, we choose a certain number of instances for one step (thus, a subset of the whole training set) and update the weights after each step. These subsets are called “mini-batches”. For example, our training set contains 1000 data instances, and we use mini-batches of size 10. This means we will have to do 100 steps for the model to “see” all the instances in the training set once and learn on them. In the **stochastic gradient descent** (SGD) setup, we use mini-batch size of 1: thus, for every step, we pick exactly one data instance and update the weights of the model based on the loss for this specific data instance. **(3 points)**

Advantages and disadvantages:

Batch gradient descent: the advantage of this method is that it guarantees that any update will be beneficial for the whole training dataset, and thus it can offer the fastest way to the optimum. The main disadvantage is that this method is not feasible in real-world scenarios, since large datasets won’t fit into memory. Also, it is prone to ending up in a local minimum.

Stochastic gradient descent: the advantage of this method is that it is not memory-hungry and easy to implement. The disadvantage is that the updates beneficial for one data point might be not so beneficial for the whole dataset (the problem of outliers), so it can take a long time to get to the optimum.

Mini-batch gradient descent: this method offers a good trade-off to address the disadvantages of the other two methods (that’s why it is most widely used in practice). One can control the size of the batches so that they fit into memory, but are still large enough to avoid updates which are not beneficial for a

significant number of data points. A minor disadvantage of this method is that in most cases it takes a bit longer to get to the optimum than if using full batch gradient descent.

(2 points)

One can say that stochastic gradient descent and mini-batch gradient descent are actually one and the same method with different parameter settings. SGD is just mini-batch with batch size =1. It does not matter much whether the students say that SGD is a subset of mini-batch training or vice versa – the important thing here is that they understand that these two methods are substantially different from the full batch gradient descent, because they use **subsets** of data instances for each update step. **(1 point)**