Einführung in Artificial Intelligence SS 2024, 4.0 VU, 192.027

Exercise Sheet 3 – Learning from Examples and Neural Networks

For the discussion part of this exercise, mark and upload your solved exercises in **TUWEL** until Wednesday, June 5, 23:55 CEST. The registration for a solution discussion ends on Friday, June 7, 23:55 CEST. Be sure that you tick only those exercises that you can solve and explain!

In the discussion, students will be asked questions about their solutions of examples they checked. The discussion will be evaluated with 0–25 points, which are weighted with the fraction of checked examples and rounded to the next integer. There is *no minimum number of points* needed for a positive grade (i.e., you do not need to participate for a positive grade, but you can get at most ≈80% without doing exercises).
Note, however, that *your registration is binding*. Thus, *if* you register for a solution discussion, then it is *mandatory* to show up. No-show after a registration without plausible excuse leads to a penalty. Such students will not have the possibility to participate in another exam once they have gotten a certificate. If you registered but cannot show up due to unpredictable and unavoidable obstacles, either deregister or send us a confirmation (doctor's note, etc) and you will be excused. Please ask questions in the **TUWEL** forum or visit our tutors during the tutor hours (see **TUWEL**).

Exercise 3.1: A close friend of you is experiencing some trouble in deciding which Lego set to buy next. You have recently learned about the wonders of artificial intelligence. In particular, how decision trees can be used to make predictions based on examples and you decide to construct a model that decides whether your friend will like a newly released Lego set or not. After a quick research about the relevant data for such a decision, you settle for the attributes P (the number of pieces in the set) with the domain $V(P) = \{<500, 500-1000, >1000\}$, R (whether the release year is after 2010) with the domain $V(R) = \{\top, \bot\}$, S (whether the set contains a lot of stickers) with the domain $V(S) = \{\top, \bot\}$ and T (theme the set belongs to) with the domain $V(T) = \{\text{star wars, superheros, technic, architecture, city, creator, ninjago}\}$. Content with this rather simplistic model, your friend provides you with the data for some sets that they have already build:

Sample	P	R	S	T	Liked?
1	$500-1000$	$\top$	$\bot$	star wars	T
2	<500	$\bot$	$\top$	ninjago	F
3	>1000	$\bot$	$\top$	technic	T
4	>1000	$\top$	$\bot$	architecture	T
5	<500	$\bot$	$\bot$	creator	T
6	$500-1000$	$\top$	$\top$	superheros	F
7	$500-1000$	$\bot$	$\top$	city	F
8	$500-1000$	$\top$	$\bot$	technic	T
9	>1000	$\bot$	$\top$	star wars	T

Use the gathered data to construct a decision tree capable of predicting whether your friend will like a set, given the values of the attributes they chose. In each step of the construction, choose the attribute that maximizes the information gain, as shown in the lecture.

Exercise 3.2: Construct another decision tree for Exercise 3.1, this time choosing the attribute that maximizes the *relative information gain*, i.e., the ratio between the gain of the attribute and its own intrinsic information.

$$GainR(A) = \frac{Gain(A)}{H(A)}$$

and

$$H(A) := \sum_{a \in V(A)} \frac{|E_a|}{S} \log_2 \frac{S}{|E_a|},$$

where $V(A)$ denotes the domain of the attribute A and $|E_a|$ is the number of all samples which have the value a of the attribute A. Furthermore, we define $S := \sum_{a \in V(A)} |E_a|$ to be the size of the set of all samples.

Exercise 3.3: In this exercise we explore some of the problems with decision trees and the ways of dealing with them:

(a) Compare the results obtained in Exercise 3.1 and Exercise 3.2. Use this example to discuss the advantages of using the relative information gain rather than the regular information gain rule. Be sure to explain *why* using the relative gain leads to better results!

(b) Discuss the design choices that were made in Exercise 3.1.

Are decision trees a suitable model for predicting which Lego sets someone will like *in practice*?

Is the choice of attributes and possible values sensible? If so, argue why, if not, provide some alternative options.

What can you say about the practical accuracy of the generated tree(s)? What could have been done to make the tree(s) more accurate (match your friend's situation better)?

(c) Suppose that you are collecting data for a decision tree. The first two examples you collect have the exact same value for each of the attributes you picked, but their classification is different. You stop collecting further data and ponder the implications of this situation. Answer the following questions briefly:

- What could be the cause of such a situation?
- What would the decision tree learning algorithm discussed in the lecture do in this case?
- What could you do to avoid the problem?

Exercise 3.4: Suppose that an attribute splits the set of examples E into k subsets $E_1, \ldots, E_k$ where each subset E_i has p_i positive and n_i negative examples. Show that the attribute has zero information gain if the ratio $\frac{p_i}{p_i + n_i}$ is equal for all $i \in \{1, \ldots, k\}$. Provide arguments for all properties that you use.

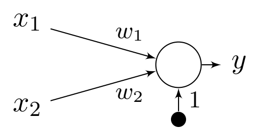
Exercise 3.5: Construct a model of a neural network with three binary input signals I_1, I_2, C_{in} and two output signals C_{out} (carry), S (sum) which represents an implementation of a *full adder*. The signals should behave according to the entries of the following truth table.

I_1	I_2	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

For each neuron, use the following activation function:

$$g(x) = \begin{cases} 1 & \text{if } x \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$$

Exercise 3.6: Consider a single layer perceptron with two input neurons and one output neuron of the following form:



Train the perceptron using the *Perceptron learning rule* and the identity activation function $g(x) = x$ on the following training data: $f(1, 2) = 2$, $f(2, 1) = 7$, $f(2, -1) = 4$ and $f(8, 2) = 7$. The weights are initialized to 0 and the learning rate α is 1. The bias weight will not be learned and stays 1.

Express the function $f(x_1, x_2)$ that your network has learned after the training is complete algebraically. Does this function produce the expected outputs for all four training inputs?

If the neuron has not learned the function correctly, provide alternative weights so that the neuron produces the expected outputs for all examples, or prove that this is impossible.

Exercise 3.7: Show formally that a single-layer perceptron with the step function as activation function ($g(x) = 1$ for $x \geq t$ for some threshold t and $g(x) = 0$ otherwise) cannot express the logical equivalence operator ($\equiv$).

Exercise 3.8:

1. Explain from a *gradient descent* viewpoint that in the multi-variable linear regression model, L_1 regularization prefers to set many weights to 0 which results in sparse models, comparing with L_2 regularization.

   Recall that the L_q regularization adds to the (empirical) loss function a regularization term $\lambda \cdot \mathit{Complexity}(h_{\mathbf{w}})$, where λ is a number and $\mathit{Complexity}(h_{\mathbf{w}}) = \sum_i |w_i|^q$, with $q \in \{1, 2\}$.

   Hint: observe the behavior of function $signum$, the derivative of absolute value function.

2. Show that the sigmoid function σ is a special softmax function with $d = 2$.

Recall the two functions as follows.

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

$$softmax(\mathbf{v})_k = \frac{e^{v_k}}{\sum_{k'}^{d} e^{v_{k'}}}, \mathbf{v} = v_1 \dots v_d.$$