Machine learning for data science I

18 June 2026

Surname, name (all caps) _____

Student ID: _____

This is a closed book exam.

Write clearly and justify your answers.

All your answers should fit on the exam pages. You can use auxiliary sheets for thinking but they should not be submitted. Only your clear final answers on the exam will be graded.

Time limit: 90 min.

Question:	1	2	3	4	Total
Points:	25	25	25	25	100
Score:					

1. A company evaluates job candidates through a sequential hiring process with the following five stages that each addresses some skills/properties of candidates: CV screening, general interview, technical interview, job/salary offer, employment. Each candidate may drop out at any stage (either due to rejection or voluntary exit). You are given a dataset of candidate features and the goal is to build a model that predicts the last stage reached by each candidate.

[5] (a) Briefly describe multinomial regression and discuss its advantages and disadvantages for this task.

[5] (b) Briefly describe ordinal regression and discuss its advantages and disadvantages for this task.

[10] (c) Propose a model that explicitly accounts for the specifics of this stage-wise hiring problem (overcoming the disadvantages of the previous two models). Clearly define the form of the model.

[5] (d) Describe how the parameters of the proposed model would be estimated from data.

2. Consider a multiclass classifier with input vector $x \in \mathbb{R}^N$, weight matrix $W \in \mathbb{R}^{M \times N}$, logits $u = Wx$, softmax outputs s_i and cross-entropy loss $L = -\sum_{i=1}^{M} y_i \log s_i$, where y is a one-hot target vector. Use i to index softmax outputs s_i, j to index logits u_j and k to index inputs x_k.

[3] (a) Draw the computational graph of this model.

[7] (b) Derive the Jacobian of the softmax function: $\frac{\partial s_i}{\partial u_j}$. Use Kronecker delta to simplify the notation ($\delta_{ij} = 1$ if $i = j$ or 0 otherwise).

[7] (c) Derive $\frac{\partial L}{\partial u_j}$ and $\frac{\partial L}{\partial W_{jk}}$.

[2] (d) How should the logit u_j change for a single example with correct class $y_j = 1$ and what can you tell about the sign of $\frac{\partial L}{\partial u_j}$?

[3] (e) Which classical machine learning model does this model resemble and how? It is over-parametrized relative to the classical formulation - what does this mean and could you remove it? Why does the neural network formulation usually keep this redundancy?

[3] (f) Why do we train such neural network models in batches of input samples and why isn't the preferred batch size 1 or N?

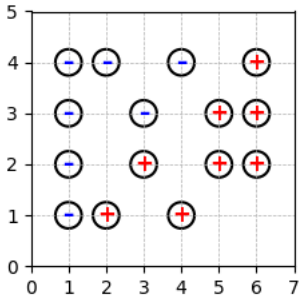
3. Answer the following question about boosting with respect to the dataset illustrated below.

[6] (a) Briefly describe the idea of boosting and the AdaBoost algorithm.

[5] (b) A typically used weak learner are axis-aligned decision stumps. You can assume that the optimal split for the dataset illustrated below is $x_1 \leq 4.5$. Compute the score of that split using the Gini impurity ($G(p) = 1 - \sum_k p_k^2$).

[9] (c) Perform 3 iterations of AdaBoost using decision stumps on the given dataset. Draw the intermediate datasets, indicate the weights and decision boundaries. You are not expected to compute exact values. Provide informed approximations and clearly justify your reasoning.

[5] (d) Draw the final decision boundary of the boosted classifier. You may assume all three weak learners achieve similar error rates in the training process. What is its classification accuracy on the training data? Indicate the confidence of the boosted model in its prediction for different regions.



4. On the next page is an implementation of multi-dimensional scaling using automatic computation of gradients and gradient descent. Note that some sections of the code marked with ___X___ are missing. Answer the following question.

[5] (a) Briefly describe MDS. What measure does it optimize?

[8] (b) The t-SNE method emphasises local relations. Which parts of t-SNE are essential for achieving this and why?

[4] (c) Write down the missing sections 1 of the MDS implementation.

[8] (d) Write down the missing sections 2 of the MDS implementation.

```python
import random
from agrad import Value


def train(model, xs, ys, learning_rate=0.001, n_epochs=1000):
    for k in range(1, n_epochs + 1):
        loss = model.loss(xs, ys)
        for p in model.parameters():
            p.grad = 0
        loss.backward()

        for p in model.parameters():
            p.data = ___1___

        if k % 100 == 0:
            loss = model.loss(xs, ys)
            print(f"{k:4} Loss: {loss.data:5.3f} {model}")
    return model


class MDS:
    def __init__(self, items):
        random.seed(100)
        self.items = items
        self.pos = {i: [Value(random.uniform(-1, 1)) for _ in range(2)]
                    for i in items}

    def __call__(self, x):
        return self.pos[x]

    def parameters(self):
        return [p for pos in self.pos.values() for p in pos]

    def loss(self, xs, ys):
        n = len(xs)
        ___2___

# input data: distances between pairs of items
distances = {("Novo Mesto", "Maribor"): 170, ("Maribor", "Koper"): 232, ...}

items = sorted(list(set(i for pair in distances.keys() for i in pair)))
pairs = list(distances.keys())
targets = [distances[p] for p in pairs]

model = MDS(items)
model = train(model, pairs, targets, n_epochs=1000, learning_rate=0.1)
```