Take-home exam
DIT867/DAT341: Applied Machine Learning, March 11–18, 2023

Course responsible: Richard Johansson, CSE (richajo@chalmers.se, +46317721887)

Formatting instructions:

- You need to submit a PDF or Word document. This should be an electronic document and not a scan of a hand-written document.

Please note:

- If there is something you don't understand about a question, please contact Richard over email or phone (9 AM – 5 PM) as soon as possible. Do not use Canvas to ask questions. No answers guaranteed between 5 PM and 9 AM.
- If you find typos or errors, please let me know and I will post an updated version as soon as I can.
- Until the submission deadline, it is **strictly prohibited to communicate with other students** about the contents of the take-home exam.
- Standard plagiarism regulations apply and the submitted file will be checked by a plagiarism detection program. Your solutions need to be your own and **you are not allowed to copy any material from any source**. (Please get in touch if you are unsure.)
- If we ask you for a technical solution, please describe **one** solution only. If you include multiple solutions, you will lose points for incorrect details in all of the solutions.

Part 1: Basic questions

You need a score of at least **21** points in this part to receive a passing grade (3).

Question 1 of 12: Classification of electrocardiogram signals (10 points)

At a hospital, we want to train a machine learning model that classifies an ECG (electrocardiogram) as normal or abnormal. At our disposal, we have already collected a large collection of ECGs, but no annotated labels are available.

We want to consider two solutions:

a) A *feature-based* solution. This approach extracts a number of features based on the shape of the ECG curve. You can assume that code for this feature extraction exists.

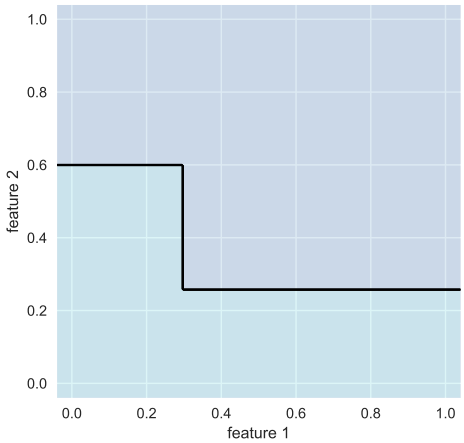
b) A *feature-free* solution that classifies the ECG signal without any feature extraction step.

Please describe the steps of the development project as a whole. You should also describe the type of ML models that you will develop in some detail for the two cases mentioned above. You can assume that this hospital dedicates a significant amount of time and resources to this project: we want the best accuracy and cost is no problem. We can assume that proper ethics processes have been followed and that we can now turn to the practical steps.

Question 2 of 12: Decision trees (5 points)

Please answer the following questions about decision tree classifiers.

(a, 1p) The figure below shows the decision boundary of a trained decision tree classifier for a binary classification task with two continuous-valued features.



Can you construct a decision tree that would give us this decision boundary?

(b, 2p) Generally, for binary decision tree classifiers using two continuous-valued features, where we set the maximally allowed tree depth to 2, please give a description of how the decision boundary can look.

(c, 1p) Do you think that our model will suffer from overfitting if we set the maximally allowed depth to 2?

(d, 1p) In our dataset, one of the features encodes a file size in bytes. We update this feature so that it is expressed in megabytes instead. How is this going to affect the training algorithm and the performance of the model? How should we modify our preprocessing?

Question 3 of 12: Problems... (4 points)

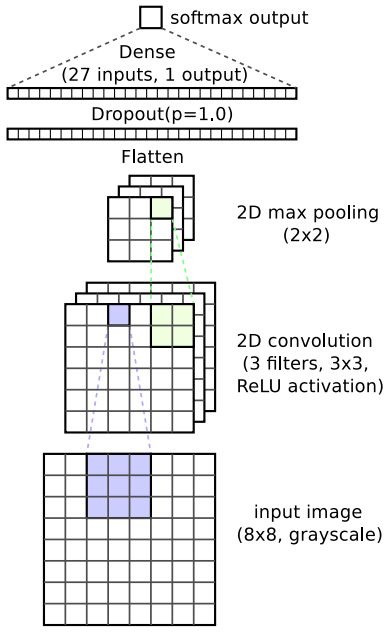
Take a look at the article *Hundreds of AI tools have been built to catch covid. None of them helped*:[1]

Go to the section *"What went wrong"* and read first few paragraphs (up to before "Errors like these"). Exemplify a hypothetical application use case from outside the medicine area where we may potentially suffer from the same issues. Explain your reasoning.

Question 4 of 12: Neural network classification and regression (4 points)

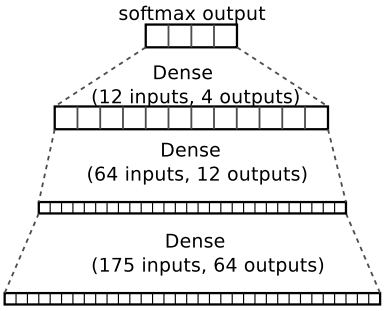
The following neural network architectures include various types of errors. Correct these errors and explain the corrections.

(a, 2p) The following convolutional neural network is designed for a binary image classification task of 8x8 grayscale images.



(b, 2p) The following model is designed to predict the market price of an apartment. It is designed as a nonlinear regression model implemented as a feedforward neural network with two hidden layers. Its input is a 175-dimensional feature vector representing various properties of the apartment (location, size, number of rooms, ...) that has been standardized before applying the regression model.

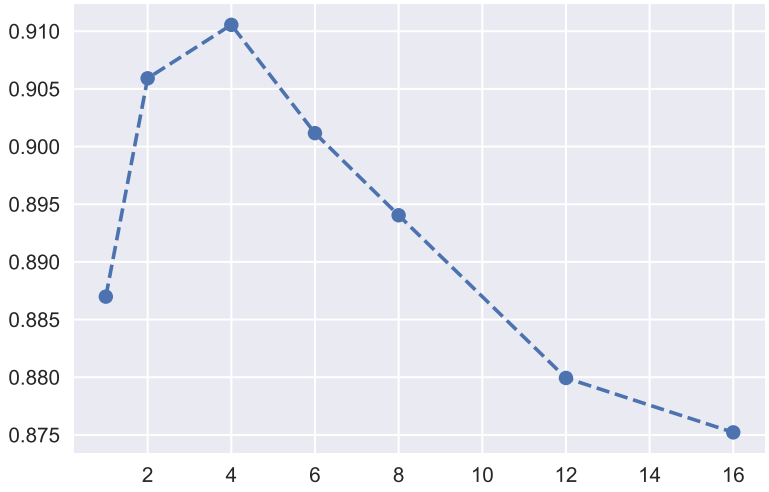
[1] https://www.technologyreview.com/2021/07/30/1030329/machine-learning-ai-failed-covid-hospital-diagnosis-pandemic



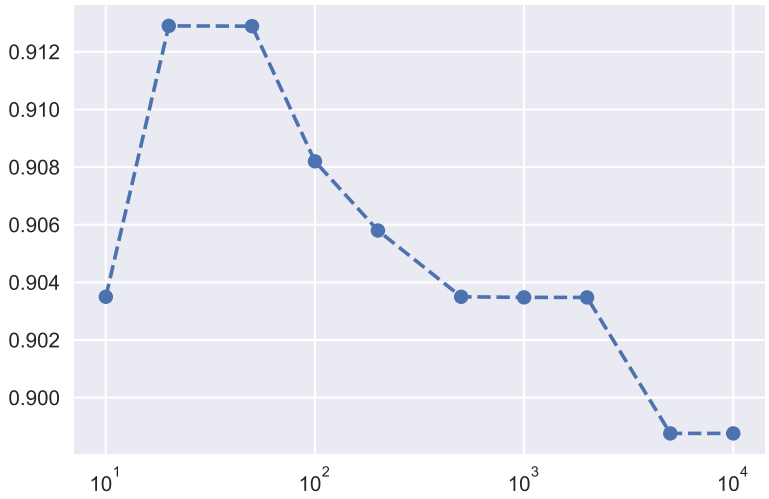
Question 5 of 12: Hyperparameter tuning in a gradient boosting classifier (4 points)

We use a `GradientBoostingClassifier` in a classification task. While tuning the hyperparameters using cross-validation, we observe the results below in (a) and (b). In both cases, explain why we see these results.

(a, 2p) `max_depth` goes from 1 to 16. All other hyperparameters are set to their default values.

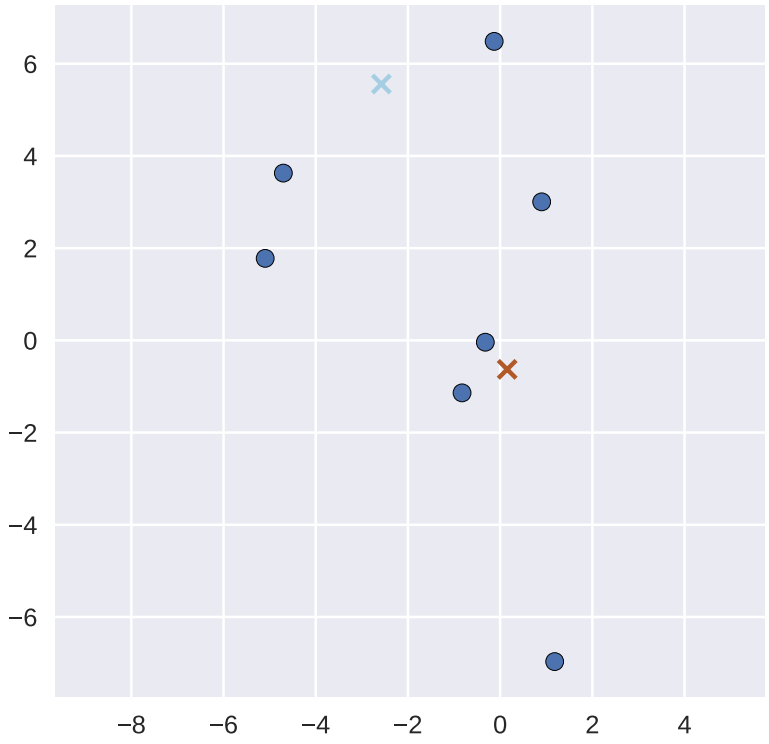


(b, 2p) `n_estimators` goes from 10 to 10,000.



Question 6 of 12: K-means clustering (5 points)

The round dots in the scatterplot below show a dataset containing 7 points. We run K-means clustering (Lloyd's algorithm) with $K = 2$. The crosses in the scatterplot correspond to the initial values of the two centroids.



(a, 3p) What centroids and cluster assignments will the algorithm converge to with this initialization? Show all the steps to reach this state.

(b, 2p) The result in (a) is not unique. For instance, if we assign the bottom right instance to its own cluster, and the remaining instances to the other cluster, we also get a stable assignment. How can we decide which of these two solutions to choose? (This question is about general principles and you are not required to compute anything here.)

Part 2: Questions for the high grades

You need a total score of **43** for the grade *4*, and **56** for the grade *5*.

Question 7 of 12: Convolutional neural networks for image classification (8 points)

(a, 2p) In a 3x3 convolutional filter for RGB-encoded images, there is a filter that is activated by blue patches of the image and is turned off where the image is not blue. In a convolutional neural network, this filter is located in the first layer – that is, it operates directly on the original image – and will be followed by a ReLU activation. Give examples of weights of such a convolutional filter.

Hint: this will be a 3x3x3 structure because of the RGB encoding of the image. The first two dimensions of this structure correspond to the horizontal and vertical dimensions, while the third is for the RGB values.

(b, 2p) We apply this filter (followed by ReLU activation) to the following image.

What is the output going to be? Describe roughly: you do not have to compute anything.

(c, 2p) We apply a 2x2 max pooling layer to the output in (b). What is the result?

(d, 2p) After applying a number of convolution and pooling layers, we produce a tensor of shape 10x10x16: that is, there are 16 feature maps are of shape 10x10. We are building a classifier that can distinguish the following categories: *city*, *desert*, *forest*, *sea*, *meadow*. Add additional layers to implement the complete classifier.

Question 8 of 12: Art recommendation (4 points)

We want to build a recommendation system for works of visual art. For each user, we have a number of training examples where we can access an image of the artwork and a user rating (1-10). Every day, new works of art are added to the system. Design a machine learning model that can give recommendations of previously unseen works of art to a given user. Describe the design of the model and how recommendations would be carried out. Describe the assumptions needed for this approach to be meaningful. You can focus on the design of the model and you do not have to describe data collection and evaluation.

Question 9 of 12: Probabilities in decision tree classifiers (6 points)

We train a decision tree classifier on a dataset consisting of continuous-valued features. In our case, we are interested in *probabilities* output by decision tree classifiers; for instance, trees in scikit-learn include a method called `predict_proba`.

There are various methods to make it possible for decision tree classifiers to predict probabilities. Recall the training algorithm for tree classifiers, also implemented in our notebook example. To extend this algorithm, instead of creating a leaf simply remembering the majority class, we create a leaf reflecting the probability distribution of the subset. For instance, if the subset of the training data used to build the leaf consists of 100 data points, 40 of which are of class A and 60 of class B, then the distribution in this leaf node becomes $P(A) = 0.4$ and $P(B) = 0.6$.

(a, 2p) When training a decision tree with scikit-learn, we set the `max_depth` hyperparameter to `None`. (See scikit-learn's documentation.) Why is this likely to give us poor probabilities? As mentioned above, the features are continuous-valued, and we can also assume that all feature values for all instances are unique.

(b, 2p) What do I actually mean when I say that the probabilities are poor? What would it mean for them not to be poor? Note that the point here is not about classification accuracy: this would not require a model that can output probabilities.

(c, 2p) Exemplify some use case where it is important that the probabilities are meaningful.

Question 10 of 12: Perceptron learning (4 points)

We can write the perceptron algorithm for training a binary classifier as follows:

Inputs: a list of example feature vectors $\boldsymbol{X}$
a list of reference outputs Y, encoded as +1 or -1
learning rate η
number of epochs N
$\boldsymbol{w}$ = zero vector of length m
repeat N times
  for each training pair $\boldsymbol{x}, y$
    if $y \cdot \boldsymbol{w} \cdot \boldsymbol{x} \leq 0$
      $\boldsymbol{w} = \boldsymbol{w} + \eta \cdot y \cdot \boldsymbol{x}$

In this pseudocode, $\boldsymbol{w}$ as usual corresponds to the model weight vector, and m is the length of the feature vectors in $\boldsymbol{X}$. After training, we classify an instance as positive if $\boldsymbol{w} \cdot \boldsymbol{x} \geq 0$, otherwise as negative.

The learning rate η is a user-defined hyperparameter and it is a constant value that is greater than 0. Discuss how the value of η affects the model $\boldsymbol{w}$ and the classification accuracy.

Question 11 of 12: Neural network regression (8 points)

We will consider a regression model implemented as a neural network with one hidden layer and ReLU activation. Expressed in matrix form, the model works as follows:

$$\begin{aligned}\boldsymbol{a} &= \boldsymbol{W}_i \cdot \boldsymbol{x} + \boldsymbol{b}_i \\ \boldsymbol{h} &= \mathrm{ReLU}(\boldsymbol{a}) \\ \mu &= \boldsymbol{w}_\mu \cdot \boldsymbol{h} + b_\mu \\ \sigma &= \exp(\boldsymbol{w}_\sigma \cdot \boldsymbol{h} + b_\sigma)\end{aligned}$$

Here, $\boldsymbol{x}$ is the input feature vector. $\boldsymbol{W}_i$ and $\boldsymbol{b}_i$ are the parameters of the input layer. The model has two parallel output heads (μ and σ) and $\boldsymbol{w}_\mu, b_\mu$ and $\boldsymbol{w}_\sigma, b_\sigma$ are the parameters of these two heads, respectively. $\boldsymbol{a}$ refers to the "pre-activation" (the input to the ReLUs) and $\boldsymbol{h}$ the hidden layer output.

The first three lines are typical of NN regression models, for instance `MLPRegressor` in scikit-learn: for a given feature vector $\boldsymbol{x}$, we compute the hidden layer output $\boldsymbol{h}$ and then the regression output μ. As usual, we can use this to predict numerical quantities, such as what we think the market value of an apartment is (as in Question 4b).

What makes this model somewhat different from a typical NN regressor is the last line, where a second output head predicts a quantity σ, which intuitively can be thought of as expressing *instance-dependent varibility*. The interpretation of the model output is that for a given input $\boldsymbol{x}$, the distribution of the y values follows a normal distribution with mean μ and standard deviation σ.

(a, 1p) Exemplify an application where you think this type of model would be more useful than a standard NN regressor.

(b, 1p) To train this model, we use the *negative log-likelihood* as the loss function. Assuming that the input feature vector is $\boldsymbol{x}$ and the observed output value is y, we compute μ and σ as above. Then the loss for this instance is

$$\text{Loss} = \log\sqrt{2\pi} + \log\sigma + \frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2$$

How is this similar to and different from the loss function used to train a `MLPRegressor`?

(c, 1p) Would it be interesting to implement a similar trick in a *classification* model (such as a `MLPClassifier` in scikit-learn) instead of a regression model?

(d, 1p) Why do you think we have an exp activation in the σ output head? Can we simplify the model and remove the exp, so that both output heads are linear?

(e, 4p) We will now train this model using a training set $\boldsymbol{X}, Y$. In this case, we will not write down the gradients explicitly here. Instead, let us assume that a deep learning library provides a function BACKWARD (similarly to the corresponding function in PyTorch). Assuming that we have computed the loss, BACKWARD(Loss, p) computes the gradient of the loss with respect to a model parameter p.

Write pseudocode to describe how we can train this model. No regularization is needed. You need to be explicit about what hyperparameters you expect the user to provide.

Question 12 of 12: Bug routing (6 points)

At a very large software development company, there are thousands of teams that work on developing various functionalities of the company's products. These teams are organized in a multi-level hierarchy: at each site, there are various departments and divisions within departments.

When bugs (software errors) are discovered, a detailed report is created. This report includes a description in free text of the circumstances of the error, often including step-by-step instructions for reproducing it and a description of its effects. Program code may be included as well, if relevant, as well as other types of semi-structured data (e.g. date of discovery, memory consumption, timing measurements, etc.)

Based on the description, each bug is then assigned to one or more departments, divisions or teams within the organization. In simple cases, just a single team needs to be involved, but if the error has more complex causes, several teams may need to cooperate.

Carrying out this assignment of bugs to teams has turned out to be cumbersome and time-consuming. For this reason, the company wants to develop a machine learning system that automatically assigns the bug to the relevant parties within the organization.

When developing this system, we should take the organizational hierarchy into account. For instance, let us assume that the correct assignment of a bug would be to to team A within division X: if the ML system accidentally predicts team B within division X, this is less severe than if we had predicted division Y, since it is less costly to re-assign incorrectly assigned bugs within the same division.

(a, 5p) Describe how you would build a bug routing system of this type. Be clear about all relevant technical and methodological details.

You can assume that there is already a suitably large annotated dataset available for a supervised approach, so you do not have to describe the data collection and annotation process. In this dataset, each bug report is associated with one or more labels corresponding to teams or higher organizational units.

(b, 1p) Suggest an application where it could be useful to have a category system as in (a), where we allow multiple labels and the labels have a hierarchical structure. Unlike the application in (a), the input to this classifier should not be a text.