Take-home exam
DIT866/DAT340: Applied Machine Learning, March 18–20, 2020

Course responsible: Richard Johansson, CSE (richard.johansson@gu.se, +46317721887)

Formatting instructions:

- You need to submit a PDF or Word document. If you prefer to write your solution on paper, it is OK to submit a scanned copy as long as you can get the scan into a PDF.

Please note:

- To keep grading anonymous, please do not include your name in the file you submit.
- If there is something you don't understand about a question, and which is not covered on the Canvas page *Frequently asked questions*, 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.)

Statement

Please include a text where you declare that

- you have not discussed with anyone (except possibly the course responsible) about the content of the exam;
- you have not copied any material from any source (printed or online).

Part 1: Basic questions

You need a score of at least **25** points in this part to receive a passing grade (*G/3*).

Question 1 of 12: Predicting fuel consumption in ships (8 points)

A shipping company would like to build a machine learning model that predicts the fuel consumption of a ship, given a number of continuous and discrete measurements such as the ship's speed, the engine's RPM (revolutions per minute), wind draft measurements, etc.

A ship travels from Southampton in the UK to Port Elizabeth in South Africa without stopping in any seaports along the way. We collect sensor readings every 5 minutes during this voyage, and also record how much fuel has been consumed during each 5-minute interval. This is a 16-day trip, so we end up with a few thousand examples in our dataset.

(a, 6p) Explain how you would implement a machine learning model for the fuel consumption prediction task. You don't need to show Python code, but please give a description of the system and explain all steps you would carry out when developing it.

(b, 2p) Discuss the limitations of the approach that we have used here.

Question 2 of 12: Training a neural network (3 points)

We use Keras to train a neural network for a binary classification task. The diagnostic output printed by Keras during training (100 epochs) is shown below.

```
Epoch   5: loss: 0.3206 - acc: 0.8499 - val_loss: 0.3289 - val_acc: 0.8460
Epoch  10: loss: 0.2962 - acc: 0.8614 - val_loss: 0.3219 - val_acc: 0.8477
Epoch  15: loss: 0.2806 - acc: 0.8695 - val_loss: 0.3216 - val_acc: 0.8502
Epoch  20: loss: 0.2681 - acc: 0.8742 - val_loss: 0.3284 - val_acc: 0.8502
Epoch  25: loss: 0.2557 - acc: 0.8812 - val_loss: 0.3311 - val_acc: 0.8498
Epoch  30: loss: 0.2448 - acc: 0.8856 - val_loss: 0.3415 - val_acc: 0.8484
Epoch  35: loss: 0.2345 - acc: 0.8907 - val_loss: 0.3516 - val_acc: 0.8465
Epoch  40: loss: 0.2250 - acc: 0.8942 - val_loss: 0.3608 - val_acc: 0.8456
Epoch  45: loss: 0.2168 - acc: 0.8985 - val_loss: 0.3766 - val_acc: 0.8428
Epoch  50: loss: 0.2098 - acc: 0.9015 - val_loss: 0.3846 - val_acc: 0.8394
Epoch  55: loss: 0.2023 - acc: 0.9052 - val_loss: 0.3946 - val_acc: 0.8380
Epoch  60: loss: 0.1950 - acc: 0.9087 - val_loss: 0.4110 - val_acc: 0.8383
Epoch  65: loss: 0.1893 - acc: 0.9113 - val_loss: 0.4182 - val_acc: 0.8386
Epoch  70: loss: 0.1827 - acc: 0.9147 - val_loss: 0.4317 - val_acc: 0.8360
Epoch  75: loss: 0.1778 - acc: 0.9175 - val_loss: 0.4427 - val_acc: 0.8338
Epoch  80: loss: 0.1714 - acc: 0.9200 - val_loss: 0.4528 - val_acc: 0.8322
Epoch  85: loss: 0.1671 - acc: 0.9222 - val_loss: 0.4641 - val_acc: 0.8322
Epoch  90: loss: 0.1626 - acc: 0.9242 - val_loss: 0.4729 - val_acc: 0.8372
Epoch  95: loss: 0.1580 - acc: 0.9263 - val_loss: 0.4855 - val_acc: 0.8317
Epoch 100: loss: 0.1538 - acc: 0.9286 - val_loss: 0.4934 - val_acc: 0.8327
```

After training, we evaluate the classifier on a separate test set and get an accuracy of 0.82. How do you think we can improve the test set accuracy somewhat?

Question 3 of 12: Blood pressure prediction (7 points)

Let's assume, unrealistically, that we have built a machine learning system that predicts the systolic blood pressure level of an individual by applying a convolutional neural network to an image of the person's face.

(a, 2p) We collect a small test set to see how the system's predictions relate to the true blood pressure values. The table below shows the result.

True value	Predicted value
131	129
142	133
105	101
147	142
120	100
90	86
114	100
145	152
138	137
101	93

How should we evaluate this system? Select an evaluation metric and compute it for this example.

(b, 2p) Define a suitable trivial baseline and evaluate it in the same way. (We don't see the training data here, but you may assume that the blood pressure measurements in the training data for this baseline are "similar" to the test data.)

(c, 3p) Let's say that we consider levels of systolic blood pressure greater than the threshold of 140 to be abnormally high (*hypertension*). If we use this system for the purpose of finding the people with abnormally high blood pressure, how should we evaluate it? Compute the relevant scores.

Question 4 of 12: Machine learning in insurance (3 points)

An insurance company develops two machine learning systems: a binary classifier that determines whether or not to offer an insurance plan to a potential customer, and a regression system that suggests a premium. The features used in these prediction systems will obviously depend on what type of insurance policy we are considering: for instance, for a car insurance we may consider the age and the traffic history of the driver, etc.

Explain why it is probably more useful for the company to use fairly small decision trees rather than random forests or neural networks when we develop these prediction systems.

Question 5 of 12: Different types of regression models (7 points)

For a regression task, let us think of how different types of models may be more or less suitable, depending on the "shape" of the dataset.

(a, 2p) What kind of dataset is ideally suited for a decision tree regression model but poorly suited for a linear model?

(b, 2p) Conversely, what kind of dataset is ideally suited for a linear regression model but poorly suited for a decision tree?

(c, 3p) How do you think neural network regression models behave with respect to the datasets you discussed in (a) and (b)?

Question 6 of 12: Machine learning in a car safety system (8 points)

In a car manufacturing company, we have developed a machine learning system that determines whether the car is skidding or not. This classifier is based on measurements read from 20 different sensors and has been trained on a large volume of historical data.

(a, 3p) After developing and training this classifier, the software in some of the sensors has been found to be faulty. The subcontractors that deliver the sensors release updates that correct these software errors. Please explain what consequences you can expect in your machine learning system and what you can do about them.

(b, 2p) In the car's safety system, our classifier is not used on its own, but is combined with two other classifiers (one based on a rule system using the same sensor values, and another on specialized hardware). The final decision is carried out by computing a weighted formula that uses the outputs from all the three classifiers. What might have motivated us to use this combination of three systems instead of just using one?

(c, 3p) Our machine learning classifier was implemented as a neural network and we have found that we can improve the accuracy slightly by switching to a tree-based gradient boosting classifier. How might this affect the combined system described in (b)?

Part 2: Questions for the high grades

DIT866: You need a total score of **63** points to receive the grade *VG*.

DAT340: You need a total score of **50** for the grade *4*, and **63** for the grade *5*.

Question 7 of 12: Transfer learning (5 points)

Describe the use of *transfer learning* in image classification problems and related tasks. How do these approaches work technically and what are the advantages? When do we think that transfer learning may or may not be applicable?

Question 8 of 12: Variations of random forests (5 points)

In decision tree models and in tree ensembles such as random forests, when the data includes *numerical* features, each tree node typically involves a comparison to a threshold value, and the left or right branch is selected depending on whether the feature's value is greater than or less than this threshold. As we have seen in lectures and assignments, in the standard decision tree learning algorithm, we find the *best* threshold for each feature by computing a homogeneity criterion with splits defined by different thresholds.

We would like to try a variant of the random forest learning algorithm, with a small twist: when we consider a feature that takes numerical values, we will use a *randomly selected* threshold instead of the *best* threshold. The threshold is selected uniformly randomly between the minimal and maximal value of the feature in the training set.

(a, 1p) Why do we end up with trees in the ensemble that are different from each other? Describe all the ways in which randomness affects the training algorithm in this case.

(b, 2p) How do you think this change will affect the learning algorithm's behavior compared to the standard random forest algorithm?

(c, 2p) Invent a novel type of random forest where you introduce randomness into the training process in some new way.

Question 9 of 12: Neural networks and related models (6 points)

Please answer the following two questions about neural networks and other types of models that are related to them.

(a, 3p) We train a binary classifier using scikit-learn as follows:

```
from sklearn.ensemble import BaggingClassifier
from sklearn.linear_model import LogisticRegression

X, Y = ... some training set ...

clf = BaggingClassifier(LogisticRegression(), n_estimators=10)
clf.fit(X, Y)
```

Show that there is a neural network that gives exactly the same output as this classifier.

Hint. Predictions in a `BaggingClassifier` are computed by averaging the probabilities computed by the component classifiers if they are probabilistic (that is, if they have a method called `predict_proba`). Otherwise, voting will be used.

(b, 3p) A *linear model tree* is a decision tree that keeps a full linear (classification or regression) model at every leaf node, in contrast to standard decision trees where the leaf nodes represent constant values.

For a neural network regression model with a one-variable input and a single hidden layer with rectified linear units, explain how to construct a linear model tree that gives the same output as the neural network for all inputs.

Question 10 of 12: Training a linear classifier (7 points)

We would like to train a binary linear classifier that tries to minimize the following loss function:

$$\text{Loss}(\boldsymbol{w}, \boldsymbol{x}, y) = \max(0, -y \cdot \boldsymbol{w} \cdot \boldsymbol{x})$$

As usual, in this formula $\boldsymbol{x}$ is a feature vector representing the instance for which we are making a prediction and $\boldsymbol{w}$ is the model's weight vector. y is a number representing the output class, coded as $+1$ or -1.

The gradient (or more precisely, a subgradient) of this loss function with respect to $\boldsymbol{w}$ is

$$\nabla_{\boldsymbol{w}}\text{Loss} = \begin{cases} -y \cdot \boldsymbol{x} & \text{if } y \cdot \boldsymbol{w} \cdot \boldsymbol{x} \leq 0 \\ (0, \ldots, 0) & \text{otherwise} \end{cases}$$

(a, 4p) Write the pseudocode (or Python approximation) for a stochastic gradient descent algorithm (with a minibatch size of 1) to train a classifier by minimizing this loss function on a training set. You can assume that the learning rate is constant and that we don't use a regularizer.

(b, 1p) Assuming your solution in (a) is correct, you have re-created a well-known machine learning algorithm. What is it called?

(c, 2p) Add an L_2 regularizer to the model. How does your pseudocode change?

Question 11 of 12: Asymmetric machine learning tasks (9 points)

In some cases, machine learning problems are "asymmetric": mistakes are more critical in some circumstances than in others. Please discuss the following questions with respect to "asymmetric" machine learning tasks.

(a, 1p) Let's say we have a classification task with categories A, B, C, ... Can you think of an application where mistakes are more dangerous for some categories than others? For instance, it is more dangerous to classify a test case as B if if the correct answer is C than vice versa?

(b, 2p) How do you think this affects your evaluation protocol?

(c, 2p) How do you think this affects your training procedure?

(d, 4p) Please discuss regression tasks in a similar fashion as you did in (a)–(c).

Question 12 of 12: Learning to play a game (8 points)

We would like to develop a machine learning system that plays a computer game: a game-playing "agent." There are different ways to train such agents, and in this task we will focus on training approaches where the system *learns from an expert*. This expert or teacher may be a human player or some other automatic game-playing system (e.g. a rule-based approach).

We will assume that this game allows a finite set of moves (such as *left*, *right*, *jump*, . . .) and that it operates in discrete time: that is, it is a "step-by-step" game.

(a, 2p) Let's first assume that we want to "learn by watching": the expert plays a number of games, and we record all game states and what the expert did in each situation. Put formally, our data consists of a set of *demonstrations* of the game $D = d_1, \ldots, d_m$, and each demonstration d_i is a sequence of game states and moves by the expert $(s_i^1, m_i^1), \ldots, (s_i^n, m_i^n)$.

How can we use these demonstrations to train a game-playing agent? Sketch an algorithm in pseudocode that describes how you would train the model.

(b, 2p) Now, let's change the training scenario so that instead of using the pre-recorded expert demonstrations, we assume that the expert is available "online" during training: for any situation in the game, we may ask the expert what is the best move under the current circumstances. Again, sketch an algorithm in pseudocode how you would train a game-playing agent.

(c, 2p) Do you see any advantages or disadvantages with the scenarios in (a) or (b)? Are they equivalent (meaning that they will generate the same training data) or not?

(d, 2p) Can you think of an application scenario that is not related to game-playing where these training approaches could be applied?