Introduction to Machine Learning (WS2020/21)

Programming Assignment 3

universität wien

Availability date: 20.12.2020
Due data: 22.1.2021

Number of tasks: 2 + 1 bonus task
Maximum achievable points: 100 + 2 bonus points

Learning Objectives

The learning objective of this third exercise is to apply techniques for dimensionality reduction as part of a supervised learning pipeline as well as deepening your understanding of decision making under uncertainty.

Submission

You have to submit both a written report and your (executable) source code. You make a submission by committing your report and source code to your gitlab repository which was created for you as part of this course (details are available in the video of the Tutorial from October 9th, 2020). Do **not** use folders or create archives of files, e.g., zip or rar. Use the following naming scheme for your files:

- The report must be a single pdf document, consisting of no more than 10 pages of size A4, using a font size of 11 pt, and 2 cm margins. Please keep the report concise—we don't expect any lengthy answers. Name your pdf report `ex3_report.pdf`.

- For each task (not each subtasks) submit a single python file name `ex3_task?.py`, where `?` is to be replaced by the task number. The python file must be executable and produce the results (plots, numbers, etc.) you use in your report.

Your code is considered as executable if it can be executed using Anaconda 3-2020.07 using only the following libraries: `numpy, scipy, sklearn, matplotlib, h5py`. **Submitting non-executable code might result in getting no points for the corresponding tasks.** Make sure (a) to highlight the beginning/end of each task/subtask in your code using comments and (b) to do the same for the output of your scripts.

You can update any made submissions as often as you want. For grading, we will consider your latest submission made before or at 22.1.2021, 23:59:59.

Questions

If you have any questions, please ask them in the respective Moodle forum. If you don't get an answer from your colleagues or us within reasonable time (48 hours), reach out to us via email:

- Lorenz Kummer, `lorenz.kummer@univie.ac.at`
- Kevin Sidak, `kevin.sidak@univie.ac.at`
- Sebastian Tschiatschek, `sebastian.tschiatschek@univie.ac.at`

Tasks

Expected runtimes where recorded on a MacBook Pro (2014) with 2.8 GHz Intel Core i5 (2 cores, 8 threads) CPU.

Task: 1: Dimensionality Reduction

(maximum achievable points: 45)

In this task you will work with the "Labeled Faces in the Wild" data set. You can use `sklearn.datasets.fetch_lfw_people` to download the data set (this function might take a while to run for the first time as it will download the data set).

Subtask 1: Data Loading and Data Preparation

(5 points)

Load the data set using the arguments `min_faces_per_person=30, resize=0.5`. Report the following:

☐ How many different people are in the data?
☐ How many images are in the data?
☐ What is the size of the images?
☐ Plot images of ten different people in the data set.

For the remaining tasks, split the data into 70% for training and 30% for validation. Use the stratified option of `sklearn.model_selection.train_test_split` (provide the class labels).

Runtime ≈ 1 second (when the data is already downloaded)

Subtask 2: Dimensionality Reduction Using PCA

(20 points)

Implement a class that implements the principal component analysis (PCA) for a given number of principal components. Do not use sklearn's or another existing implementation but implement PCA yourself (of course you can use `numpy` functions for computing eigenvalues / the SVD, etc.). The class should enable to compute the PCA, to project given data onto the principal components, and to reconstruct images from the projection onto the principal components.

☐ Briefly describe your implementation in the report.

Use your PCA class to fit projections onto $d \in \{5, 10, 20, 40, 80, 160, 320, 640\}$ principal components from your training data.

☐ Plot the first 5 principal components as images.
☐ Visualize 10 reconstructed images for each d.

Train a logistic regression classifier (you can use sklearn's implementation) using the projected data for each d (use a `MinMaxScaler` to normalize the data; you might need to increase the number of iterations of the solver for LogisticRegression using the `max_iter` argument).

☐ Report the achieved classification accuracy of the classifiers on the training and validation set for all d.
☐ Comment on your observations.

Runtime ≈ 1.5 minutes

Hint: You can verify correctness of your PCA implementation by comparing its outputs with that of sklearn's PCA.

Subtask 3: Dimensionality Reduction Using Autoencoders

(20 points)

Use sklearn to train autoencoders for dimensionality reduction. In particular use the `sklearn.neural_network.MLPRegressor` class with hidden sizes of (a, b, d, b, a) for $a, b \in \mathbb{N}, d \in \{40, 80\}$ and rectified linear units. You can then use the following code snippets to encode images (in vector form) and decode images from compressed representations (where `mlp` denotes the MLPRegressor instance):

```
def encode(X, mlp):
    """
    This function is not working for general MLPs,
    the MLP must have the layer-configuration as
    stated in the exercise description.

    X must have the shape
    n_images x (witdh in pixels * height in pixels)
```

```
    """
    z = X
    for i in range(len(mlp.coefs_) // 2):
        z = z @ mlp.coefs_[i] + mlp.intercepts_[i]
        z = np.maximum(z, 0)

    return z

def decode(Z, mlp):
    """
    This function is not working for general MLPs,
    the MLP must have the layer-configuration as
    stated in the exercise description.

    Z must have the shape n_images x d
    """
    z = Z
    for i in range(len(mlp.coefs_) // 2, len(mlp.coefs_)):
        z = z @ mlp.coefs_[i] + mlp.intercepts_[i]
        z = np.maximum(z, 0)

    return z
```

Use your Autoencoder to project the images from the training data onto $d \in \{40, 80\}$ dimensions. Try reasonable choices for the hidden layer sizes (see also the hint below).

☐ Visualize 10 reconstructed images for each d.

Train a logistic regression classifier (you can use sklearn's implementation) using the projected data for each d (use a `MinMaxScaler` to normalize the data; you might need to increase the number of iterations of the solver for LogisticRegression using the `max_iter` argument).

☐ Report the achieved classification accuracy of the classifiers on the training and validation set for all d.
☐ Comment on your observations.
☐ Compare your results to those of the previous task.

Runtime ≈ 60 *minutes for training two models with* $a = 500, b = 100$ *and* 300 *epochs of training*

Hint: It is sufficient to try two different architectures and report results for those. You could start from the configuration mentioned above when commenting on the runtime.

Task: 2: Decision Making Under Uncertainty

(maximum achievable points: 55)

It's Christmas time and Santa has many presents to deliver. He can't deliver presents to all houses but has to selected a subset of all houses to optimize certain objectives (see below). Presents can only be efficiently delivered through chimneys and it is much more time consuming to deliver them in other (magical) ways. Unfortunately, in recent years, more and more houses without chimneys are built, making the planning of the delivery process more and more challenging. Santa's objective is to, in some subtasks, in particular deliver presents to children. Luckily, we can leverage ideas from machine learning and decision theory to support the planning.

The data for this task is available at [1] and has the following features:

- x and y coordinates of a house (dimensions 0 and 1)
- age a of a house dimension (2)
- average carbon-dioxide emission c at the house (dimension 3)
- distance to next school (dimension 4)
- average number of children living in houses in the neighborhood (dimension 5)

The training data also comes with the following labels:

- House has a chimney (dimension 0)
  ("true"/"false" encoded as "1"/"0")
- Are children living in the house (dimension 1)
  ("true"/"false" encoded as "1"/"0")

Subtask 1: Kernelized Logistic Regression

(20 points)

Implement regularized kernelized logistic regression. Your implementation should accept a kernel function which allows to compute the kernel function (for fast execution, this kernel function should ideally support the efficient computation of the kernel matrix for multiple data points in a single call). For optimization, feel free to use the Adam optimizer.

☐ Briefly describe how you implement learning and prediction.

Evaluate and test your kernelized logistic regression function on the regression data available from [1] using an RBF kernel computed only on the geographic distance between data points. Once you have downloaded the data and stored in a local folder, you can load the data as follows:

```
import h5py
hf = h5py.File('regression.h5', 'r')
x_train = np.array(hf.get('x_train'))
y_train = np.array(hf.get('y_train'))
x_test = np.array(hf.get('x_test'))
y_test = np.array(hf.get('y_test'))
hf.close()
```

You might want to standardize the scale of the data if that improves performance.

☐ Report classification accuracy on the training and validation data.

Runtime is less than 1 *s for* 100 *iterations of gradient descent (including kernel computation, etc. but no hyperparameter tuning)*

Subtask 2: Improving your kernel

(10 points)

Improve the performance of kernelized logistic regression by using more features from the data in a customized kernel.

☐ Describe the design of your kernel.

If you need to choose hyper-parameters, use some variant for hyper-parameter search and cross-validation to do so.

☐ Report the classification accuracy of your improved kernel on the training and validation data and the selected hyper-parameters (if any).

Runtime $\approx$ *35 minutes for 1458 candidate hyper-parameter settings and 5-fold cross-validation (training and testing 7290 models)*

Subtask 3: Supporting Planning I

(10 points)

Assume there is a limit (constraint) $k = 50$ on the number of houses Santa can visit (and assume that only the number of houses is a constraint; other aspects like travelled distance, etc. do not matter). Furthermore, for planning the delivery, Santa can only access the features describing houses but not whether they actually have a chimney or not. Santa considers the following costs:

$$c(a, \text{chimney}) = \begin{cases} 0 & a = \text{deliver}, \text{chimney} = 1 \\ 10 & a = \text{deliver}, \text{chimney} = 0 \\ 1 & a = \text{don't deliver}, \text{chimney} = 1 \\ 1 & a = \text{don't deliver}, \text{chimney} = 0 \end{cases} \tag{1}$$

☐ What objective should Santa optimize to minimize the expected costs? Simplify the objective as much as possible (similar as in the lecture).
☐ Implement this strategy and report the expected costs on the validation data.
☐ Compare the achieved costs with that obtained by a strategy randomly picking k locations (average over 50 random selections).
☐ How many houses with children does Santa deliver presents to?

Runtime $\approx$ 1 second

Subtask 4: Supporting Planning II

(15 points)

Assume Santa wants to focus his present delivery on houses with children. In particular, he considers the following cost function:

$$c(a, \text{chimney}, \text{children}) = \begin{cases} 0 & a = \text{deliver}, \text{chimney} = 1, \text{children} = 1 \\ 5 & a = \text{deliver}, \text{chimney} = 1, \text{children} = 0 \\ 2 & a = \text{deliver}, \text{chimney} = 0, \text{children} = 1 \\ 10 & a = \text{deliver}, \text{chimney} = 0, \text{children} = 0 \\ 20 & a = \text{don't deliver}, \text{chimney} = 1, \text{children} = 1 \\ 10 & a = \text{don't deliver}, \text{chimney} = 1, \text{children} = 0 \\ 10 & a = \text{don't deliver}, \text{chimney} = 0, \text{children} = 1 \\ 5 & a = \text{don't deliver}, \text{chimney} = 0, \text{children} = 0 \end{cases}$$

Train a regularized kernelized logistic regression model to predict the number of children in a house. Assume that this prediction is independent of the prediction on whether a house has a chimney or not.

☐ Derive and report the Bayes' optimal decision of whether Santa should deliver presents to a house or not.

Assume there is a limit (constraint) $k = 50$ on the number of houses Santa can visit (and assume that only the number of houses is a constraint; other aspects like travelled distance, etc. do not matter). Furthermore, for planning the delivery, Santa can only access the features describing houses but not whether they actually have a chimney or not or whether children live in the house or not.

☐ What objective should Santa optimize to minimize expected costs?
☐ Implement this strategy and report the expected reward on the validation data.
☐ Compare the the achieved reward with that obtained by strategy randomly picking k locations (average over 50 random selections).
☐ How many houses with children does Santa deliver presents to?

Runtime $\approx$ 10 seconds (no tuning of the model for predicting the number of children)

Task: 3: Trial exam

(maximum achievable points: 2 bonus points)

Subtask 1: Uploading an exam

(2 bonus points)

Upload a hand-written and scanned/photographed page showing your name and the solution to the algebraic problem stated on the assignment. You can find the "trial exam" and the upload page on Moodle in the section "Trial Exam". Please solve this exercise using the equipment you will be using when solving the actual exam on January 29.

References

[1] *Regression Dataset*. URL: https://tschiatschek.net/course/IML/WS2020/exercise3/task2/regression.h5.