Introduction to Machine Learning (WS2020/21)

Programming Assignment 1

Availability date: 9.10.2020
Due data: 16.11.2020

Number of tasks: 5
Maximum achievable points: 100

Learning Objectives

The learning objective of this first exercise is to get some first hands on experience in implementing, training and using basic machine learning models and approaches. You will familiarize yourself with the supervised machine learning pipeline, approaches for evaluating machine learning models and customize machine learning models to specific applications.

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 `ex1_report.pdf`.
- For each task (not each subtasks) submit a single python file name `ex1_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`.
You can update any made submissions as often as you want. For grading, we will consider your latest submission made before or at 16.11.2020, 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

In all tasks you are required to write Python scripts to solve the tasks. You can use all functions available in scikit-learn to solve the tasks unless stated otherwise. When splitting data into training and test data, make sure that the data is shuffled before it is split.

Task: 1: Least Squares Linear Regression

(maximum achievable points: 20)

In this task you will work with the "California Housing" data set [5].

Subtask 1: Data Preparation

(2 points)

Load the "California Housing" data set (`sklearn.datasets.fetch_california_housing`). Split the data into a training set consisting of 70% of the samples and a test set of 30% of the samples (`sklearn.model_selection.train_test_s`
Use this split of the data for the remainder of this exercise. Report the number of features of the data set, and the number of samples in the training data and in the test data.

Subtask 2: Regression Using scikit-learn

(2 points)

Train a least squares linear regression model (without bias; `sklearn.linear_model.LinearRegression`) to fit the training data. Report the MSE (`sklearn.metrics.mean_squared_error`) of this model on the training and the test set.

Subtask 3: Implement Linear Regression on Your Own I

(3 points)

Repeat the previous task but this time do not use scikit-learn's `LinearRegression` class for fitting the training data and making predictions but rather implement the closed form solution for least squares linear regression as discussed in the lecture yourself and use the resulting weight vector for making predictions. Report your computed weight vector. Again, compute the MSE on the training and test set for the obtained regressor and report these results. Do the results match your results from before?

Subtask 4: Implement Least Squares Linear Regression on Your Own II

(5 points)

Repeat the previous task but this time do not implement the closed form solution for least squares linear regression but implement gradient descent, i.e., implement the gradient of the loss function and the parameter update. Start your training from an all-zero weight vector. Use a learning rate schedule of $\frac{10^{-6}}{1+t}$, where t is t-th iteration of gradient descent. Generate plots showing the MSE on the training and test set (on the y-axis) after $1 + 1000k$ gradient steps, where $k \in \{0, \ldots, 100\}$ (x-axis). Describe what you observe in the plots. How do the final MSEs compare to that of the close-form solution?

Subtask 5: Implement Least Squares Linear Regression on Your Own III

(3 points)

Implement an adaptive learning rate rule of your choice and demonstrate that it leads to faster learning wrt minimizing the MSE. Report the same plots as above.

Subtask 6: Implement Least Squares Linear Regression on Your Own IV

(3 points)

Repeat the previous two tasks but scale the data using the `sklearn.preprocessing.MinMaxScaler` before. Report your findings (plots + description). Describe how the `MinMaxScaler` transforms the data (i.e., report some equation).

Subtask 7: Least Squares Linear Regression Using Higher-order Features

(2 points)

Scale the data using the `sklearn.preprocessing.MinMaxScaler`. Then, generate polynomial features (`sklearn.preprocessing.PolynomialFeatures`) of degrees 2, 3, and 4 of the data and fit least squares linear regression models for each of these degrees. Report the MSEs on the training and test data for each polynomial degree. What do you observe?

Task: 2: Non-linear Regression and Classification

(maximum achievable points: 20)

Subtask 1: Toy Data Regression

(9 points)

Download the "Toy Regression" dataset from [1]. Once stored in a local folder, you can load data as follows:

```
import h5py
hf = h5py.File('toy-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()
```

Construct non-linear features of the data such that a least squares linear regression model fitted on the training data achieves an MSE below 0.01 on the test data. The features you construct must not be polynomial features. Describe your approach. What is the best performance you achieve using polynomial features?
Hint: visualize the data to get insights into a useful non-linear features.

Subtask 2: Toy Data Classification

(11 points)

Download the "Toy Classification" dataset from [2]. Once stored in a local folder, you can load the data similarly as described above. Construct non-linear features such that a linear SVM (`sklearn.svm.LinearSCV`) fitted on the training data achieves 95% accuracy on the test data. The features you construct must not be polynomial features. Compute and report the confusion matrix (`sklearn.metrics.confusion_matrix`) for the test data . Describe your approach and your observations.
Hint: visualize the data to get insights into useful non-linear features.

Task: 3: Estimating Generalization Errors

(maximum achievable points: 20)

In this task you will work with the "California Housing" data set [5]. Split the data into a training set (30% of the data) and a test set (70% of the data) and keep this split fixed for the whole task. Normalize the data using the `MinMaxScaler`.

Subtask 1: Baseline

(1 points)

Fit a least squares linear regression model to the training data. Compute the MSE on the training and test data and report it. Compute polynomial features of degree 2 of the data. Again, compute the MSE on the training and test data and report it. What do you observe?

Subtask 2: Implement Cross-validation

(8 points)

Implement k-fold cross-validation for estimating the MSE of a least squares linear predictor. Do not use scikit-learn's implementation for cross-validation but rather implement cross-validation on your own. Estimate the MSE of the linear predictor using cross-validation with $k = 10$ on the training data. What MSE do you observe? How does this performance compare to the MSE on the test set from the previous subtask?

Subtask 3: Model Selection

(11 points)

Train least squares linear regression models for different degrees of polynomial features $p \in \{1, 2, 3, 4, 5\}$ of the data and use the following three methods to choose the optimal degree wrt prediction performance:

(1) Select the best degree with respect to the MSE on the training data.

(2) Split the training data into new training data (50% of the data, chosen randomly) and validation data (the remaining 50% of the data). Train linear regression models on the new (smaller) training set and pick the best degree with respect to the MSE on the validation data.

(3) Use 10-fold cross-validation for estimating the MSE of least squares linear regression models for different degrees of polynomial features.

Evaluate the linear regression model with the found degrees on the test data and report your results. Do the approaches yield consistent results regarding the "optimal" degree? If they differ, why do they differ?

Task: 4: Feature Selection

(maximum achievable points: 20)

In this task you will use the wine dataset [6]. Load (`sklearn.datasets.load_wine`) the data and normalize it using the `MinMaxScaler`. Split the data into 70% training data and 30% test data. As the classification model use the `sklearn.svm.LinearSVC` with the default parameters.

Subtask 1: Forward Greedy Feature Selection

(10 points)

Implement forward greedy feature selection wrt the 10-fold cross-validation performance. Report the order of the features (including their names) for forward and backward selection. Also record and report the performance of models using each number of features.

Subtask 2: Backward Greedy Feature Selection

(8 points)

Repeat the previous task but implement backward greedy feature selection instead.

Subtask 3: Feature Importance

(2 points)

Assume that you are only allowed to use 4 out of the 13 features for classification. Which features would you select using forward and backward feature selection? Do the features differ and if so, why?

Task: 5: Kernelized SVM

(maximum achievable points: 20)

In this task you will work with the "20 newsgroups" data set [4]. Load the training and testing data (`sklearn.datasets.fetch_20newsgroups_vectorized` and/or `sklearn.datasets.fetch_20newsgroups`). Note that the dataset is already partitioned into training and test data and that there is a parameter for loading the training/test data. Use a SVM (`sklearn.svm.SVC`) for classification and accuracy as metric (`sklearn.metrics.accuracy_score`).

Subtask 1: SVMs With Different Kernels

(5 points)

Use the vectorized version of the dataset (`datasets.fetch_20newsgroups_vectorized`). Evaluate a support vector machine (using default parameters except for the kernel) on the dataset, using different kernels (linear, poly, rbf, sigmoid). Select good hyperparameters for the kernels using model selection techniques. Report the classification accuracies on the training and test set and how much model selection improved performance.

Subtask 2: Develop Your Own Kernel

(15 points)

Develop a custom kernel that achieves higher accuracy as compared to your results from the previous subtask (see [3]) for an example of how to do this). You can either base your kernel on the vectorized version of the data set or use the non-vectorized version. Describe your kernel and the intuition behind your kernel. Proof that you actually implemented a kernel. If you fail, describe what you tried/thought about.
Hint: familiarize yourself with the classification task. Don't work with all samples during the development of your kernel but rather use a subset of the data.

Useful scikit-learn Functions

You might find the following list of scikit-learn functions/classes useful for solving the tasks in this assignment:

- `sklearn.datasets.fetch_california_housing`
- `sklearn.model_selection.train_test_split`
- `sklearn.linear_model.LinearRegression`
- `sklearn.metrics.mean_squared_error`
- `sklearn.preprocessing.MinMaxScaler`
- `sklearn.preprocessing.PolynomialFeatures`
- `sklearn.svm.LinearSVC`
- `sklearn.metrics.confusion_matrix`
- `sklearn.metrics.accuracy_score`
- `sklearn.model_selection.cross_val_score`
- `sklearn.datasets.load_wine`
- `sklearn.datasets.fetch_20newsgroups`
- `sklearn.datasets.fetch_20newsgroups_vectorized`
- `sklearn.svm.SVC`

References

[1] *A toy dataset for regression.* URL: `https://tschiatschek.net/course/IML/WS2020/exercise1/task2/toy-regression.h5`.

[2] *A toy dataset for regression.* URL: `https://tschiatschek.net/course/IML/WS2020/exercise1/task2/toy-classification.h5`.

[3] *Custom kernels in scikit-learn.* URL: `%5Curl%7Bhttps://scikit-learn.org/stable/auto_examples/svm/plot_custom_kernel.html%7D`.

[4] *scikit-learn documentation for the "20 Newsgroups" data set.* URL: `https://scikit-learn.org/0.19/datasets/twenty%5C_newsgroups.html%5C#newsgroups`.

[5] *scikit-learn documentation for the "California Housing" dataset.* URL: `https://scikit-learn.org/stable/datasets/index.html%5C#california-housing-dataset`.

[6] *scikit-learn documentation for the "Wine" data set.* URL: `https://scikit-learn.org/stable/datasets/index.html#wine-dataset`.