M916 - Machine Learning
Coursework 1

Due Date: 31 March, 2022

Introduction

- This coursework is an introduction to the basics of Machine Learning. It is highly recommend to take your time and understand the fundamental calculus and rationale behind the introduced approaches and avoid searching ready-made solutions online.
- Individual Coursework.
- Total marks: 100. This coursework will contribute 20% of the final grade.
- Must be submitted via e-class. E-mailed submissions will not be accepted.
- Programming:
  - For the programming part, you can turn to StackOverflow if you bump into some kind of bug or problem, but again I strongly advise against copying code for these simple tasks.
  - As mentioned in the first tutorial, it is best practice to use Virtual Environments for your projects to avoid breaking the system. You can use either Virtual Environments of native python venv or install an Anaconda/Miniconda platform. **Or simply use Google Colab**.
  - Implement your approach using a Jupyter Notebook, with sufficient but not redundant comments.

1 Vector and Matrices (10 points)

Tasks

Use the NumPy python package to generate:

1. Two random integer matrices $\mathbf{X} \in \mathbb{Z}^{3\times 4}$, $Y \in \mathbb{Z}^{4\times 3}$.
2. Two random integer vectors $\mathbf{g} \in \mathbb{Z}^4$ and $\mathbf{z} \in \mathbb{Z}^4$.

For the generation of the matrices you can use the numpy.random.

1. Compute the inner product of $\mathbf{g}$ and $\mathbf{z}$, $\mathbf{g}^T\mathbf{z} \in \mathbb{Z}$.
2. Compute the matrix-vector product $\mathbf{Xg} \in \mathbb{Z}^3$.
3. Compute the matrix or dot product $\mathbf{XY} \in \mathbb{Z}^{3\times 3}$.
4. Compute the L_2/Frobenius norms of the vectors and matrices respectively.

In all cases perform the computations by hand, showing the intermediate steps and confirm the results with a NumPy implementation. You'll need to hand in both the handwritten notes and the python notebooks.

NOTE: Set the seed to be the last digit of your student number to get consistent random results.

2 Gradient Computation (10 points)

Recall the gradient of a function $f : \mathbb{R}^n \to \mathbb{R}$, i.e., $\nabla f(x)$ with respect to a vector $x \in \mathbb{R}^n$:

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix} \tag{1}$$

Tasks

1. Let $f(x) = \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{b}^T \mathbf{x}$, where $\mathbf{A}$ is a $n \times n$ symmetric matrix such that $\mathbf{A}^T = \mathbf{A}$ and $\mathbf{b} \in \mathbb{R}^n$. Compute the gradient of $f(x)$, $\nabla f(x)$ with respect to $\mathbf{x}$ by hand showing the intermediate steps.
2. Chain rule: Assume that we have three functions, f, g and h, such that $f(x) = \sin x$, $g(x) = x^3$ and $h(x) = 5x$. Compute the derivative of $c(x) = (f \circ g \circ h)(x) = f(g(h(x)))$ by hand showing the intermediate steps.

3 Linear Regression (80 Points)

In the second lecture, we have showed how to obtain the optimal coefficients β for both the uni-variate and the multi-variate case. This was done by setting the derivatives with respect to β of the loss function to zero.

3.1 Derivation of the Least Square estimator for the multivariate case (multiple regressors) (5 points)

In Linear Regression, we assume a dataset containing m pairs of observations $\{\mathbf{x}_i, y_i\}_{i=1}^m$, where $\mathbf{x}_i$ is a $p-$dimensional vector of regressors. We seek to derive an estimator/linear predictor $h(\cdot)$ to predict y given $\mathbf{X}$, such that:

$$h(\mathbf{X}) = \beta^T \mathbf{X} \tag{2}$$

β-vector contains the intercept/bias factor in its vector notation.

Task:

- Derive the Least Squares estimates for the multivariate Linear Regression case. **(5 points)**

Show the intermediate steps of the computations.

3.2 Implementation (75 Points)

In this task, you'll be using python and specifically *NumPy, Scikit-learn, Matplotlib* and *Pandas* to explore Linear Regression in a common introductory dataset: The Boston Housing Dataset.

Pandas is a python package for data manipulation and analysis. It provides some very useful functions and makes data access easier and more intuitive. There are many tutorials online about the usage, but the main object you'll be using is a pandas Dataframe.

Dataset Overview

The dataset consists of 506 samples and 13 features/regressors. We want to use this data in order to construct an estimator to be able to predict the price of a house given the features. You can load the dataset and create a single panda dataframe for both the dataset data and the target variable.

The features of the Boston House Dataset are:

- CRIM: Per capita crime rate by town.
- ZN: Proportion of residential land zoned for lots over 25,000 sq. ft.
- INDUS: Proportion of non-retail business acres per town.
- CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).
- NOX: Nitric oxide concentration (parts per 10 million).
- RM: Average number of rooms per dwelling.
- AGE: Proportion of owner-occupied units built prior to 1940.
- DIS: Weighted distances to five Boston employment centers.
- RAD: Index of accessibility to radial highways.
- TAX: Full-value property tax rate per 10, 000.
- PTRATIO: Pupil-teacher ratio by town.
- B: 1000(Bk — 0.63)^2, where Bk is the proportion of [people of African American descent] by town.
- LSTAT: Percentage of lower status of the population.

Target Variable (y): MEDV: Median value of owner-occupied homes in $1000's.

Preliminaries

In Machine Learning, it is bad practice to use the training data to evaluate the model. Hence, we need to split the available data into two different subsets: the *training* and the *test* data. There exist implementations that perform this task, and in this exercise you can use the train_test_split function of the sklearn package to do that. Essentially you train the model using the train data and evaluate it on both the training and test data.

Tasks:

- **Train the model (60 Points)**:
  1. Use the train_test_split sklearn function to split the data into train and test sets using a ratio of your choice, e.g. 90% training, 10% test, e.t.c.
  2. Train a linear regression model on the training data using the normal equations. Do not use the built in implementation of the sklearn package, but instead employ the coefficient estimates that you derived in the previous section.
- **Evaluate the results (15 points):**
  1. Evaluate the model, using the statistics introduced in Linear regression lecture for both training and test data (RSE, R^2).
  2. Is there a difference between the produced RSE and R^2 statistics of the train and test sets? If yes, why do you think this happens?
  3. Choose the following training/test ratios, 80/20, 70/30, 50/50 for train_test_split and repeat the training process. Show the corresponding RSE and R^2 metrics for each split and explain the potential differences among the obtained estimators.