Lecture 2: Assignment 02
Linear Regression
EECE-6822 Machine Learning

Cris Ababei
Electrical and Computer Engr., Marquette University

1. Objective
The objectives of this activity include: (1) run several given code examples that deal with regression in order to get better insights; and (2) modify some of the examples to apply regression to other problems.

2. Prerequisite Readings
Murphy
- Linear algebra review: Murphy 7.1-7.3
- Matrix calculus review: Murphy 7.8
- Maximum likelihood regression: Murphy 4.2
- Linear regression; Linear regression with basis functions; Cross-validation: Murphy 11-11.2

Mostafa
- The Linear Model: Ch. 3

James-Hastie
- Ch.3 Linear Regression

Geron
- Ch. 2 presents an end-to-end ML project – super nice exposure to most of the steps we usually go when working on a project. It focuses on example of house prices in California. Uses notebooks we can run in Google colab. Uses a basic LinearRegression as a model, then a DecisionTreeRegressor. You should read and at the same time run the code.
- Ch.4 (Linear Regression Section, up to Gradient Descent) presents a brief presentation of linear regression equations; emphasis is on the **Normal Equation** (which is just given; for a derivation of that please see hand-written notes). The normal equation gives a closed-form solution for theta_hat; basically the solution to the minimization of MSE problem. Then, the LineraRegression of SciKit-Learn is used to show that it provides similar result as the example code using the Normal Equation.
- Ch.4 (Polynomial Regression Section, up to Logistic Regression) talks about polynomial regression, ridge regression (adds regularization term to MSE cost function), Lasso regression (also adds regularization term to MSE cost function, but uses l1 norm of the weight vector, not l2), Elastic Net regression. It also talks about overfitting; early stopping.

Raschka – Ch.9
- Discussion of regression models to predict target variables on a continuous scale. It talks about exploring and visualizing datasets. Several approaches to implement linear regression models. Training regression models to be robust to outliers. Fitting regression models to nonlinear data.

3. Examples

Example 1: Regression - NumPy

Open your google colab and upload the notebook, provided with the files for this lecture:
code/linear_regression_demo.ipynb

This notebook illustrates linear regression on the **diabetes dataset**:

- 10 explanatory variables of each patient (age, sex, bmi, bp, s1,s2,s3,s4,s5,s6)
- outcome y is measure of diabetes progression over 1 year
- we want to predict y given those 10 features

It uses **numpy.linalg.lstsq**.
Take some time and read about it at (the provided notebook build on the example provided here):
https://numpy.org/doc/2.1/reference/generated/numpy.linalg.lstsq.html

Go through all the steps of the notebook. Pause and search/learn about all the things you encounter that are unclear to you.

Example 2: Linear Regression with Basis Functions

Open your google colab and upload the notebook, provided with the files for this lecture:
code/demo_polynomial.ipynb
This notebook illustrates linear regression on a **synthetic train dataset with n=100 points**, where ground truth is 5-th order polynomial, which has added Gaussian noise to it. The test dataset is also with 100 points and generated randomly with the same polynomial.
It investigates linear regression with a constant fit, affine fit, and polynomial features with p=2,3,4,5,10,15,20. We will see that as we continued to increase the size of our polynomial, the test error started to go up.

The main takeaways from this example are:

- Know the difference between underfitting (model is not complex enough to even fit the training data) and overfitting (model is too complex and begins to fit the random idiosyncracies of the training data in a way that will not generalize to test data)
- Diagnose the above by comparing train error and validation error
- Feature engineering benefits from domain knowledge about what are good features for the application. Or at least carefully inspecting model fit

Example 3: Regression - SciKit-Learn

In this activity, we use code examples from:
[*B3-Geron] Aurelien Geron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, O'Reilly, 2022.
The source code of this book is in this GitHub repository:
https://github.com/ageron/handson-ml3
This is an excellent book which emphasizes hands-on examples. Its first part focuses on SciKit-Learn and then in the second part on Keras and Tensorflow.

The example we look at is from Chapter 4. The corresponding notebook is at:
https://github.com/ageron/handson-ml3/blob/main/04_training_linear_models.ipynb
Launch that in your google colab and go through the steps **up to and including Linear Regression**.

In this notebook, regression is done on a **small array dataset (size m=100) with randomly generated** numbers. Try to understand the code; pay attention to the discussion and implementation of the Normal Equation. Then, pay particular attention the use of LinearRegression class **from sklearn.linear_model.** Note that **LinearRegression** class is based on the **scipy.linalg.lstsq()** function (the name stands for "least squares"), which was also used in the example code from Part 1 of this Assignment.

Example 4: Regression – example in the context of an end-to-end ML project

In this activity, we again use code examples from:
[*B3-Geron] Aurelien Geron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, O'Reilly, 2022.

Here, you should read and at the same time run the code from Ch. 2, which presents an end-to-end ML project – super nice exposure to most of the steps we usually go when working on a project. It focuses on an example **dataset of house prices in California**. Focus your learning on the use of a basic **LinearRegression** as a model, then a DecisionTreeRegressor.

Example 5: Regression – SciKit-Learn

This activity uses code example from:
[*B3-Raschka] Sebastian Raschka, Yuxi Liu, and Vahid Mirjalili, Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python, Packt Publishing, 2022.

This is another excellent book which also emphasizes hands-on examples. However, it focuses on implementing models and algorithms in an object orient approach. It relies on SciKit-Learn (used in the firsr part of Geron book as well), but, then it focuses on PyTorch (as opposed to Keras and Tensorflow used in Geron book). ***Another nice thing about Raschka book is that it presents examples both as notebooks and Python code, which you can run (and especially easier modify) in Anaconda Spyder for example.*** *Not to mention that its PDF is free too. Finally, notations used in the book are similar/consistent with those used for example by Jason Brownlee of mlm (meachinelearningmastery.com).*
The source code of this book is in this GitHub repository:
https://github.com/rasbt/machine-learning-book

NOTES:
Alternative ways to run code examples from Raschka book, which provides Python code directly in addition to notebooks:

(1)Approach #1: Open and run Jupyter notebooks locally on your computer
First, download directly the entire GitHub repository. Extract in a folder on your computer and always work there, with either the Jupiter Notebooks or the Python scripts. For example, I have downloaded and unzipped it as:
EECE6822\Book_Code_Raschka\machine-learning-book-main
To launch a Jupyter Notebook – such as:
EECE6822\Book_Code_Raschka\machine-learning-book-main\ch02\ch02.ipynb
You can:

- Navigate to the location of the notebook in File Explorer. Right-click on it and select Open with -> Choose another app. Navigate to C:\anaconda3\Scripts and select **jupyter-notebook.exe**.
- In Anaconda Navigator, locate the Jupyter Notebook tile and click "Launch".
- Open an Anaconda Prompt and type jupyter notebook. This will open the Jupyter Notebook interface in your web browser, allowing you to create or open notebooks.

(2)Approach #2: Open and run Python code (same as in notebook) locally on your computer
To run the equivalent Python script – such as:
EECE6822\Book_Code_Raschka\machine-learning-book-main\ch02\ch02.py

Launch Anaconda Spyder and open the script. **This is my preferred way as it is the easiest to make changes to provided code.**

(3)Approach #3: Open and run Jupyter notebook in google colab

- Follow the directions here for how to do that: https://github.com/rasbt/machine-learning-book/blob/main/supplementary/running-on-colab.pdf
- Or take the link from github of the notebook you want to run. For example: https://github.com/rasbt/machine-learning-book/blob/main/ch02/ch02.ipynb
  Open it in google colab.
  Note that if you want to run it, you will have issues for examples with importing figures/
  To address that, create a code-box at the top of the notebook with the following code.
  Code-box 1:

```
!git clone --depth=1 https://github.com/rasbt/machine-learning-book.git
%cd /content/machine-learning-book
import sys, os
sys.path.append("/content/machine-learning-book")
os.chdir("/content/machine-learning-book/ch02")
```

  Then, just run the notebook as usual.

NOTE:
If you work with conda environments, follow the steps from Chapter 1 of the book text. Also explained here: https://github.com/rasbt/machine-learning-book/blob/main/ch01/README.md

At this time, open or upload into Google Colab the notebook for Ch. 9, located at:
https://github.com/rasbt/machine-learning-book/blob/main/ch09/ch09.ipynb
and go through all the steps concomitantly with reading the chapter.

The examples in this chapter are applied on the **Ames Housing Dataset**. This chapter packs a lot, but, focus your attention to the linear regression concept: simple versus multiple. First, **observe the implementation of class LinearRegressionGD** and its use on the dataset.
Then, see how SciKit-Learn library can be used for developing working models for regression analysis – which can be more efficient than the straightforward LinearRegressionGD class presented earlier.
Next, the notebook presents the use on RANSAC (random sample consensus), which fits a regression model to a subset of data, the so-called inliers. It then presents a discussion of evaluating performance of linear regression models.

At this point, you could stop here or continue with the rest of the notebook: using regularized methods for regression: Regularization is one approach to tackling the problem of overfitting by adding additional information and thereby shrinking the parameter values of the model to induce a penalty against complexity.
Then, study polynomial regression, Random forests, and Decision tree regression – for dealing with nonlinear relationships.

Note that we will re-visit this code example of Ch.9 later when we'll talk about gradient descent.

Example 6: Regression – Direct Implementation in Python (mlm)

In this part, we will look at several tutorials from **machinelearningmastery (mml)**, which has several tutorials on regression:
https://machinelearningmastery.com/?s=linear+regression

Specifically:
---Read:
A Gentle Introduction to Linear Regression with Maximum Likelihood Estimation
https://machinelearningmastery.com/linear-regression-with-maximum-likelihood-estimation/

Some key points:

"There are many ways to estimate the parameters in linear regression. There are two frameworks that are the most common: (1) Least Squares Optimization. (2) Maximum Likelihood Estimation.

Both are optimization procedures that involve searching for different model parameters.

Least squares optimization is an approach to estimating the parameters of a model by seeking a set of parameters that results in the smallest squared error between the predictions of the model (yhat) and the actual outputs (y), averaged over all examples in the dataset, so-called mean squared error.
Maximum Likelihood Estimation is a frequentist probabilistic framework that seeks a set of parameters for the model that maximize a likelihood function. We look at this second approach."

---Read:
How to Solve Linear Regression Using Linear Algebra
https://machinelearningmastery.com/solve-linear-regression-using-linear-algebra/
You will look at:

- Matrix Formulation of Linear Regression
- Linear Regression Dataset
- Solve Directly in Python and then make predictions

Notice the "Matrix Formulation of Linear Regression", which was discussed in class!

Note also (and take time to follow the links for further reading) the API section at the end of this tutorial:

numpy.linalg.inv() API

numpy.linalg.qr() API

numpy.linalg.svd() API

numpy.diag() API

numpy.linalg.pinv() API

numpy.linalg.lstsq() API

---Read:
How to Implement Linear Regression From Scratch in Python
https://machinelearningmastery.com/implement-linear-regression-stochastic-gradient-descent-scratch-python/
You will discover how to implement stochastic gradient descent to optimize a linear regression algorithm from scratch with Python. This tutorial teaches how:

- How to estimate linear regression coefficients using stochastic gradient descent.
- How to make predictions for multivariate linear regression.

Linear regression algorithm with stochastic gradient descent is uset to model the wine quality dataset.

---Read:
How To Implement Simple Linear Regression From Scratch With Python
https://machinelearningmastery.com/implement-simple-linear-regression-scratch-python/

Discover how to implement the **simple** linear regression algorithm from scratch in Python.

It uses statistics on the training data to estimate the coefficients required by the model to make predictions on new data.

You will know: How to estimate statistical quantities from training data.
Dataset: Swedish Insurance Dataset is used.

4. Assignment

--(a)Create a new Jupyter notebook (or work directly as a Python program) to redo Example #1, but for a different dataset: Boston house-price data. This is a dataset taken from the StatLib library which is maintained at Carnegie Mellon University. The dataset has 506 samples. Each datapoint or sample has 13 attributes or Features Xi (i from 0 to 12) of houses at different locations around the Boston suburbs in the late 1970s. The attributes themselves are defined in the StatLib website (as per capita crime rate in the area, number of rooms, distance from employment center, etc.). The target (Y) is the median values of the houses at a location (in USD 1,000).
The dataset is available online at: https://lib.stat.cmu.edu/datasets/boston
You can also find it in tensorflow:

```
data = tf.keras.datasets.boston_housing
(x_train, y_train), (x_test, y_test) = data.load_data()
```

--(b)Copy the needed code from Example #1 to build a regression model for prediction of house prices given 13 input features from the Boston house-price data.

--(c)Present a discussion of your findings.

5. Deliverables

You must write (typed) a report and upload it as a PDF file on D2L. The report should be named "**hw2_report_LastName.pdf**". You should also create a .zip archive with all your code and implementations of all parts of the assignment. Upload also this archive .zip file with the name "**hw2_implementation_code_LastName.zip**" to D2L. Hence, your D2L should contain two items: the report and the .zip file. **Do not include the report inside the .zip and upload only the .zip. They should be two separate items!**

The report should include the following sections and subsections:

1) Title + course info + your name
2) **Summary.** Describe in one paragraph what the objective of the assignment is.
3) **Description.** Describe the regression model implemented and studied. Include all the figures you created and discuss them in your report. All plots in your figures should have axes labels, and all figures should be numbered and have short captions describing what the figures present.
4) **Conclusion.** Present your conclusions; highlight what are your main takeaways that you learned from this lecture and assignment. Describe what issues you encountered and how you solved them.
5) **References.** Include all references that you used, as a numbered list. Cite them in the report itself; do not just list them! If your report has References that are not cited in the report, points will be deducted!