THE UNIVERSITY OF WESTERN AUSTRALIA

CITS5508: Assignment 1

Semester 1, 2025

Assessed, worth **20%**. Due **11:59 pm Friday 18th April 2025**.

1 Outline

This assignment is divided into two parts: a *regression* and a *classification* task. Detailed descriptions of each task are given below.

While this assignment is centred around programming, you should treat your submission as a report. A specific set of tasks is given for both **Part 1**, and **Part 2** however, your assignment is to perform an investigation. Do not expect to receive full marks for one-word answers when asked to comment on your results. Explain your thoughts in detail.

2 Submission

- You should submit your assignment via LMS. The submission should be a single Jupyter Notebook file (with an `.ipynb` extension).
- Your notebook should include all code, outputs, descriptions and comments. Treat it as a report – introduce the topic, describe your methodology, present your results and conclusions.
  - Make use of Markdown cells inserted at appropriate places to explain your code. In addition to Markdown cells, some short comments can be put alongside the Python code
  - Dividing the assignment into suitable sections and subsections (with section and subsection numbers and meaningful headings) would make your portfolio easier to follow.
  - Please see `sample.ipynb` under the Week 2 content on LMS for an example of how to structure your notebook.
- Your notebook should already include all output, but must also run without errors when using the CITS5508 Anaconda environment. Note that Google Colaboratory will produce identical outputs to our environment.

3 Part 1 – Ridge Regression

Your first task is investigate the performance of two methods for fitting Ridge Regression: the **closed-form solution**, and **Stochastic Gradient Descent** (SGD).

We will define the Ridge Regression cost function as,

$$\mathcal{L}(\mathbf{w}) = ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \alpha||\mathbf{w}||_2^2. \tag{1}$$

You may assume that $\mathbf{X} \in \mathbb{R}^{n \times m}$, $\mathbf{w} \in \mathbb{R}^m$, and $\mathbf{y} \in \mathbb{R}^n$. Then, the closed-form solution that minimises $\mathcal{L}(\mathbf{w})$ is given by,

$$\hat{\mathbf{w}} = (\mathbf{X}^\intercal\mathbf{X} + \alpha I_m)^{-1}\mathbf{X}^\intercal y, \tag{2}$$

where I_m is the $m \times m$ identity matrix,

$$I_m = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}.$$

Note that you will want to avoid regularising the bias term. The easiest way to do this is to use the bias-trick, and then you can simply set $I_{0,0} = 0$.

3.1 Tasks

1. Write a function that can generate a random toy n-degree polynomial dataset of the form,

$$y = \sum_{k=0}^{n} a_k x^k + \epsilon, \tag{3}$$

   where $x \in [-5, 5]$, $a_k \sim \mathcal{U}(0, 1)$, and $\epsilon \sim \mathcal{N}(0, 1)$. Your function should take two arguments: `degree`, and `n_samples`. Show your function works by plotting some generated datasets.

2. Implement Ridge Regression using the closed-form solution provided above with vectorised NumPy code.

3. Generate three toy datasets using your function: a linear, a quadratic, and cubic dataset. Use `n_sample = 1000` to keep the problem small for now.

4. Fit your implementation and scikit-learn's Ridge Regression via `sklearn.linear_model.SGDRegressor` on your toy datasets. Don't forget to add polynomial features when needed. Try $\alpha = \{0, 0.1, 100\}$ and compare the two implementations.

5. Comment on your results in detail. Investigate the following ideas:

   (a) Comment on your choice of parameters for `sklearn.linear_model.SGDRegressor`.

   (b) Create scatter plots of your toy datasets. Plot your model and `SGDRegressor`. Compare $\mathbf{w}$ between the implementations.

   (c) Is your implementation and scikit-learn's producing the same output? Does α affect both models how you would expect?

6. Experiment with higher degree polynomials and larger values of `n_samples`. Comment on your results in detail. Investigate the following ideas:

   (a) How does your implementation compare to `SGDRegressor`?

   (b) Which implementation do you expect to be faster to fit? What about for making predictions? Can you observe this on your datasets?

4 Part 2 – Support Vector Classifier

Your second task is to build a **Support Vector Machine** classification model from computer-derived nuclear features to distinguish malignant from benign breast cytology based on the *Diagnostic Wisconsin Breast Cancer Database*[1]. The dataset can be downloaded from the UC Irvine Machine Learning Repository via:

https://doi.org/10.24432/C5DW2B

Please download the dataset and place the extracted files in the same directory as your notebook. Do not use the "import in Python" option from the website.

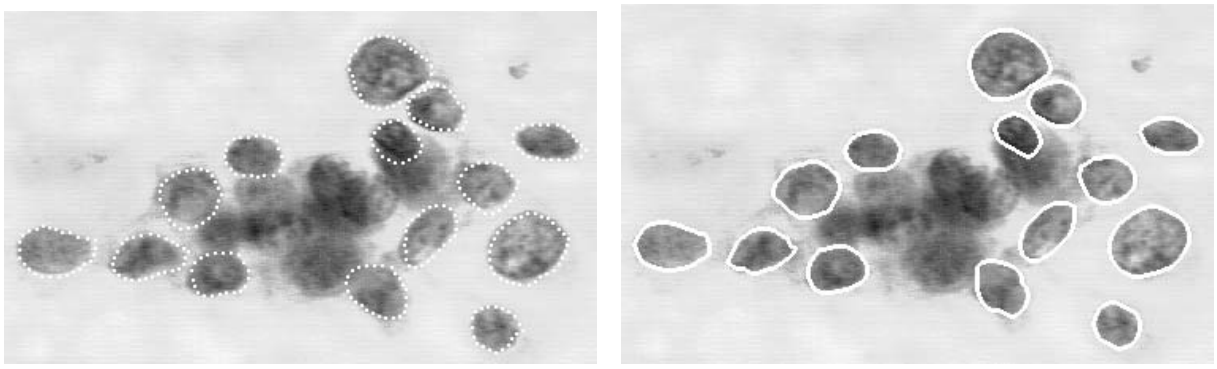

Figure 1: Visualisation of steps to extract features from the breast tissue. This part has been done for you by the original authors. Your task is to process the extracted features.

We will treat this as an end-to-end machine learning project. Before starting, you should read the included information from `wdbc.names` to understand the dataset. It may also be a good idea to read the original paper by Street *et al.* (1993)[1] to understand the context, although this is not strictly necessary.

4.1 Tasks

1. Load the dataset. Make sure all columns are named correctly. Read `wdbc.names` for information on how the columns are named.
2. Do basic exploratory data analysis. Visualise the data, and drop any unnecessary columns.
3. Split the dataset into a training and testing set. Use 80% of the data for training, and 20% for testing. Do you need to stratify your split?
4. Perform grid-search with k-fold cross-validation ($k = 5$) on at least two hyperparameter with three different values each ($3^2 = 9$ total combinations). Use a `sklearn.svm.SVC` model.
   (a) Do you need to stratify your cross-validation?
   (b) What scoring metric is most appropriate for this task?
   (c) What hyperparameters did you choose? How many models did you fit in total?
   (d) Present your results for the optimal hyperparameter combination.
5. Given your optimal model found in the previous step, evaluate your model on the test set. Present your results.
6. Discuss your results and whether the model is suitable this task. Comment on the following:
   (a) Would you be confident in implementing this for real-world breast cancer screening? Explain your reasoning.
   (b) What about precision/recall curves? Would they be useful for this task? Could you implement them for your model? Explain your reasoning.

[1] Street, W., W. Wolberg, and O. Mangasarian. 'Breast Cancer Wisconsin (Diagnostic)'. UCI Machine Learning Repository, 1993. https://doi.org/10.24432/C5DW2B.