CSE 4/574: Introduction to Machine Learning, Sections A, C & D

Spring 2023

Instructor: Alina Vereshchaka

Assignment 1

Classification and Regression Methods

Checkpoint: Feb 23, Thu, 11:59 pm

Due Date: Mar 9, Thu, 11:59 pm

Description

Our second assignment is focused on learning how to build classification and regression models from scratch. In the first part of the assignment, you will implement logistic regression and test it on a Penguin dataset. In the second part, you will implement linear regression using ordinary least squares and in the third part, you will extend it to ridge regression. In addition to these, you have two optional bonus components where you have to build Gradient descent and Elastic Net Regularization from scratch.

The goal of this assignment is to practice implementing logistic classification and understanding the practical implementation of different regression models. Apply these methods to solve the tasks based on the real-world datasets.

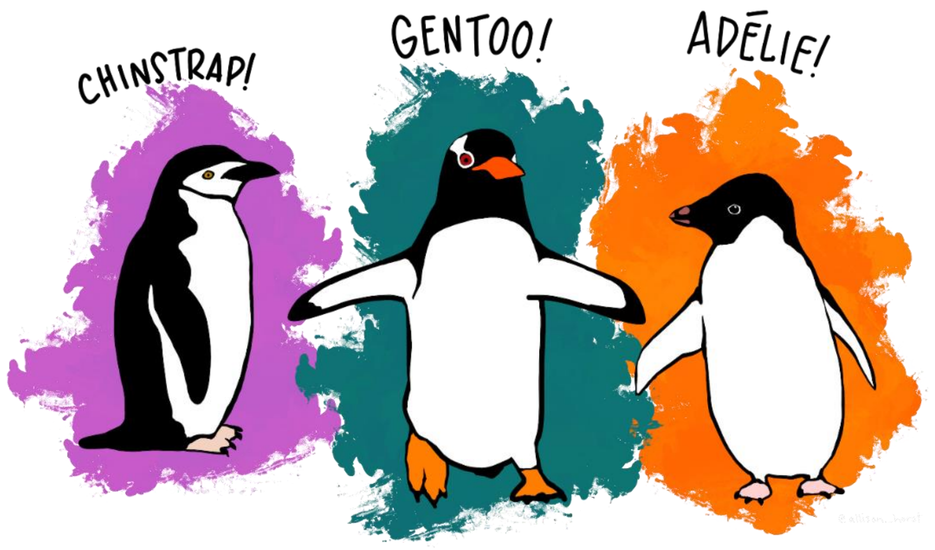
For this assignment, scikit-learn or any other libraries with in-built functions that help to implement ML methods cannot be used. Submissions with used ML libraries (e.g. scikit-learn) will not be evaluated.

Part I: Logistic Regression [40 points]

In this part, we will work on logistic regression and will use a logistic function to model a binomial (Binary / Bernoulli) output variable. Logistic regression model predicts that the observation belongs to a particular category. To generate these probabilities, logistic regression uses the sigmoid function. This function maps a real number to a value between 0 and 1.

DATASET

For this Part, we will use Palmer Archipelago (Antarctica) penguin dataset. It contains three penguin species and includes measurements of bill length, bill depth, flipper length and body mass. Overall, we are provided with 344 data samples.



Artwork by @allison_horst

The dataset consists of 7 columns:

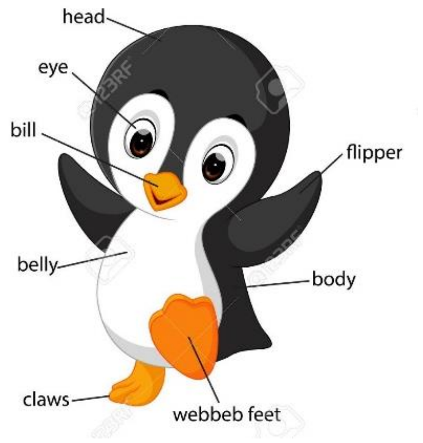
- **species**: penguin species (Chinstrap, Adélie, or Gentoo)
- **bill_length_mm**: culmen length (mm)
- **bill_depth_mm**: culmen depth (mm)
- **flipper_length_mm**: flipper length (mm)
- **body_mass_g**: body mass (g)
- **island**: island name (Dream, Torgersen, or Biscoe) in the Palmer Archipelago (Antarctica)
- **sex**: penguin sex (female, male)

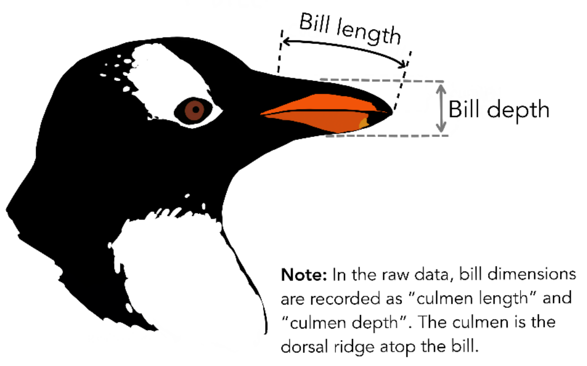
What are culmen length & depth?

The culmen is "the upper ridge of a bird's beak"

What are flippers?

Penguins' wings are called flippers. They are flat, thin, and broad with a long, tapered shape and a blunt, rounded tip.





Note: In the raw data, bill dimensions are recorded as "culmen length" and "culmen depth". The culmen is the dorsal ridge atop the bill.

Image source

STEPS

1. Import required libraries (not allowed: scikit-learn or any other libraries with in-built functions that help to implement ML methods).
2. Read, preprocess, and print the main statistics about the dataset (you can reuse your code from Assignment 0 with a proper citation)
3. Convert features with string datatype to categorical (species, island, sex).

   Example: suppose you have a dataset that contains information about movies, with the following features: title (string), director (string), genre (string). You need to convert these features of string datatype to categorical features. This can be done by assigning a unique numerical value to each unique string value in each categorical feature.
4. Normalize non-categorical features (bill_length_mm, bill_depth_mm, flipper_length_mm, body_mass_g**).**
   a. Find the min and max values for each column.
   b. Rescale dataset columns to the range from 0 to 1

   Why do we do this? Normalization is to transform features to be on a similar scale. This improves the performance and training stability of the model.

   Note: normalize() is not allowed as it is a part of scikit-learn library.
5. Choose your target Y. For this dataset, there are several options:
   a. We can use a binary classifier to predict which gender a penguin belongs to (female or male). In this case, column sex can be used as Y (target)
   b. We can use a binary classifier to predict if a penguin's location is Torgersen island or not. In this case, column island can be used as Y (target)
6. Create the data matrices for X (input) and Y (target) in a shape,$\mathrm{X} = N \times d$ and $\mathrm{Y} = N \times 1$, were N is a number of data samples and d has a number of features.
7. Divide the dataset into training and test, as 80% training, 20% testing dataset.
8. Print the shape of your X_train, y_train, X_test, y_test
9. Recommended structure of your code to define logistic regression:

```
class LogitRegression()

  def __init__()
     # Takes as an input hyperparameters: learning rate and the
     number of iterations.

  def sigmoid():
     # Define a sigmoid function as
```

$$\sigma(z) \ = \frac{1}{1+e^{-z}}$$

```
def cost():
   # Loss function for Logistic Regression can be defined as
```

$$h = \sigma(w^T x + b) = \left(\frac{1}{1+e^{-(w^T x+b)}}\right)$$

$$J(w) = \frac{1}{N}(-y * \log(h) - (1-y) * \log{(1-h)}$$

```
def gradient_descent():
   # Define current prediction y_hat for logistic regression as
```

$$\hat{y} \ = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{X} + \mathrm{b})$$

```
   # Gradient descent is just the derivative of the loss
   function with respect to its weights. Thus
```

$$\frac{\partial J(w)}{\partial w} = \frac{1}{N} X^T (\sigma(w^T x + b) - y)$$

```
   To implement this formula, you can use intermediate
   variables, e.g.
         pred = $\sigma(w^T x + b)$
         delta = pred – y_train
         dW = ($X^T$*delta)/N

def fit():
   # This method performs the training.
   # Initialize weights
   # For a number of iterations
         # Call gradient_descent function
         # Call cost function and keep it in an array , e.g.
         loss.append()

def predict(self, X):
   # Return the predicted result in the binary form
```

$$\hat{y} = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{X} + \mathrm{b}) = \begin{cases} +1 & \text{if } \sigma(z) \geq 0.5 \\ 0 & \text{if } \sigma(z) < 0.5 \end{cases}$$

10. Train the model:
    a. Define a model by calling LogitRegression class and passing hyperparameters, e.g.

       `model = LogitRegression(learning_rate, iterations)`

    b. Train the model, by calling fit function and passing your training dataset, e.g

       `model.fit(X_train, y_train)`

c. Suggested hyperparameters:

   Note: You can try different learning rates and number of iterations to improve your accuracy (accuracy of greater than 64% is expected)

   ```
   learning_rate=1e-3
   ```

   ```
   iterations=100000
   ```

   ```
   weights = np.random.uniform(0, 1)
   ```

d. Save the weights of the model, that returns the highest accuracy as pickle file. You will need to submit it along with your other files. ([Check more details about using Pickle]())

11. Make a prediction on test dataset by counting how many correct/incorrect predictions your model makes and print your accuracy
12. Print out the loss values over each iteration

You can make binary (e.g., male/female) or multiclass (e.g. Dream/Torgersen/Biscoe) classifications.

Accepted accuracy rate for penguin dataset is above 64%.

In your report for Part I:

1. Provide your best accuracy.
2. Include loss graph and provide a short analysis of the results.
3. Explain how hyperparameters influence the accuracy of the model. Provide at least 3 different setups with learning rate and #iterations and discuss the results along with plotting of graphs.

   Ex: For 3 different learning rates, you can plot the graph to discuss impact and loss over the iterations.
4. Discuss the benefits/drawbacks of using a Logistic Regression model.

Checkpoint Submission (Part I):

- **[Optional for the Checkpoint submission]** Report (as a pdf file): named as TEAMMATE1_TEAMMATE2 _assignment1_report.pdf
  e.g., avereshc_ pinazmoh_ assignment1_report.pdf
- Code (as ipynb file with saved outputs) named as

  TEAMMATE1_TEAMMATE2 _assignment1_part_1.ipynb
  e.g., avereshc_ pinazmoh_ assignment1_ part_1.ipynb
- Pickle files with saved weights that generate the best results for your model named as

TEAMMATE1_TEAMMATE2 _assignment1_part1.pickle

Part II: Linear Regression [30 points]

In this part, we implement linear regression model and apply this model to solve a problem based on the real-world dataset.

Datasets that can be used for this part (provided in the zip folder):

- Flight price prediction dataset
- Breeding Bird Atlas
- Diamond dataset (Note: x, y and z columns refer to length, width, and depth respectively)
- Emissions by Country dataset
- Epicurious – Recipes with Rating and nutrition

Implement linear regression using the ordinary least squares (OLS) method to perform direct minimization of the squared loss function.

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{N}(y_i - w^T x_i)^2$$

In matrix-vector notation, the loss function can be written as:

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})$$

where $\boldsymbol{X}$ is the input data matrix, $\boldsymbol{y}$ is the target vector, and $\boldsymbol{w}$ is the weight vector for regression.

STEPS

1. Select one dataset from the list provided above. The datasets are located in the folder “dataset”, use only the dataset provided in the folder.
2. Import required libraries (not allowed: scikit-learn or any other libraries with in-built functions that help to implement ML).
3. Read, preprocess and print the main statistic about the dataset (your code from Part I can be reused).
4. Using any data visualization library (e.g. matplotlib, seaborn, plotly), provide at least 5 visualization graphs related to your dataset. You can utilize any columns or a combination of columns in your dataset to generate graphs. E.g. correlation matrix, features vs. the target, counts of categorical features vs. the target.
5. Convert features with string datatype to categorical and normalize non-categorical features if needed.

6. Choose your target Y.
7. Create the data matrices for X (input) and Y (target) in a shape $\mathrm{X} = N \times d$ and $\mathrm{Y} = N \times 1$, where N is a number of data samples and d is a number of features.
8. Divide the dataset into training and test, as 80% training, 20% testing dataset.
9. Print the shape of your X_train, y_train, X_test, y_test.
10. Calculate the weights with the OLS equation:
$$\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$
11. Get the predictions and calculating the sum of squared errors:
$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \boldsymbol{w}^T x_i)^2$$
12. Plot the predictions vs the actual targets.

In your report for Part II:

1. Provide brief details about the nature of your dataset. What is it about? What type of data are we encountering? How many entries and variables does the dataset comprise?
2. Provide the main statistics about the entries of the dataset (mean, std, number of missing values, etc.)
3. Provide at least 5 visualization graphs with a brief description for each graph, e.g. discuss if there are any interesting patterns or correlations.
4. Provide your loss value
5. Show the plot comparing the predictions vs the actual test data
6. Discuss the benefits/drawbacks of using OLS estimate for computing the weights
7. Discuss the benefits/drawbacks of using a Linear Regression model.

Part III: Ridge Regression [30 points]

Use your implementation from Part II and extend it to Ridge Regression.

Implement parameter estimation for ridge regression by minimizing the regularized squared loss as follows:

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{N}(y_i - w^T x_i)^2 + \frac{1}{2}\lambda \boldsymbol{w}^T\boldsymbol{w}$$

In matrix-vector notation, the squared loss can be written as:

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) + \frac{1}{2}\lambda\boldsymbol{w}^T\boldsymbol{w}$$

OLS equation for Ridge regression can be estimated as

$$\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{\lambda I})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

STEPS

Reuse steps from Part II while using OLS equation for Ridge regression to learn the parameters.

In your report for Part III:

1. Provide your loss value.
2. Show the plot comparing the predictions vs the actual test data.
3. Discuss the difference between Linear and Ridge regressions. What is the main motivation for using l2 regularization?
4. Discuss the benefits/drawbacks of using a Ridge Regression model.

Final Submission (reupload your Part I and completed Part II &Part III):

- Report (as a pdf file): combine the reports for all parts into one pdf file named as
  TEAMMATE1_TEAMMATE2 _assignment1_report.pdf
  (e.g., avereshc_ pinazmoh_ assignment1_report.pdf)
- Code (as ipynb file with saved outputs): separate file for Part I and a single file for both Part II & III named as

  TEAMMATE1_TEAMMATE2 _assignment1_part_1.ipynb
  TEAMMATE1_TEAMMATE2 _assignment1_part_2_3.ipynb
  (e.g., avereshc_ pinazmoh_ assignment1_ part_1.ipynb)
- Pickle files with saved weights that generate the best results for your model for Part II & Part III named as
  TEAMMATE1_TEAMMATE2 _assignment1_part1.pickle
  TEAMMATE1_TEAMMATE2 _assignment1_part2.pickle
  TEAMMATE1_TEAMMATE2 _assignment1_part3.pickle

- Notes:
  - Ensure that your code follows a clear structure and contains comments for the main functions and specific attributes related to your solution. You can submit multiple files, but they all need to be labeled with a clear name.
  - After executing command python main.py in the first level directory or Jupyter Notebook, it should generate all the results and plots you used in your report and print them out in a clear manner.

Bonus points [max 12 points]

Gradient Descent from Scratch [5 points]

Based on your implementation of Ridge Regression in Part III, implement a gradient descent method from scratch. Discuss the results. Provide training and testing errors and time to train.

Elastic Net Regularization from Scratch [7 points]

Based on your implementation of Ridge regression in Part III, implement an elastic net regularization and compare the results with Part III.

Bonus part submission:

- Create a separate Jupyter Notebook (.ipynb) named as TEAMMATE1_TEAMMATE2 _assignment1_bonus.ipynb e.g., avereshc_ pinazmoh_ assignment1_bonus.ipynb
- You can duplicate code from your Part III if needed
- Submit Jupyter Notebook (.ipynb) with saved outputs
- A file with saved weights that generate the best results for your model (.pickle).
- Report is not required, you can include all the analysis as part of your Jupeter Notebook.

ASSIGNMENT STEPS

1. Register your team (February 17)

You may work individually or in a team of up to 2 people. The evaluation will be the same for a team of any size.

Register your team at UBLearns (UBlearns > Tools > Groups). In case you joined the wrong group, make a private post on piazza.

2. Submit checkpoint (February 23)

- Complete Part I of the assignment
- For the checkpoint report for Part I is not mandatory, you can complete the report and submit it along with the final submission

- Add all your assignment files in a zip folder including .ipynb files, the report (if available) and .pickle file at UBLearns > Assignments
- Name zip folder with all the files as TEAMMATE1_TEAMMATE2 _assignment1_checkpoint.zip
  e.g., avereshc_ pinazmoh_ assignment1_checkpoint.zip
- Submit to UBLearns > Assignments
- If you are working in a team, we expect equal contribution for the assignment. Each team member is expected to make a code-related contribution. Provide a contribution summary by each team member in the form of a table below. If the contribution is highly skewed, then the scores of the team members may be scaled w.r.t the contribution.

Team Member	Assignment Part	Contribution (%)

3. Submit final results (March 9)

- Fully complete all parts of the assignment
- Submit to UBLearns > Assignments
- Add all your assignment files in a zip folder including ipynb files for Part I, Part II, Part III & Bonus part (optional), the report and .pickle file at UBLearns > Assignments
- Name zip folder with all the files as TEAMMATE1_TEAMMATE2 _assignment1_final.zip
  e.g. avereshc_ pinazmoh_ assignment1_final.zip
- Your Jupyter notebook should be saved with the results. If you are submitting python scripts, after extracting the ZIP file and executing command python main.py in the first level directory, all the generated results and plots you used in your report should appear printed out in a clear manner.
- Include all the references that have been used to complete the assignment.
- If you are working in a team, we expect equal contribution for the assignment. Each team member is expected to make a code-related contribution. Provide a contribution summary by each team member in the form of a table below. If the contribution is highly skewed, then the scores of the team members may be scaled w.r.t the contribution.

Team Member	Assignment Part	Contribution (%)

Academic Integrity

The standing policy of the Department is that all students involved in any academic integrity violation (e.g., plagiarism in any way, shape, or form) will receive an F grade for the course. The catalog describes plagiarism as “Copying or receiving material from any source and submitting that material as one’s own, without acknowledging and citing the particular debts to the source, or in any other manner representing the work of another as one’s own.”. Refer to the Office of Academic Integrity for more details.

Important Information

This project can be done in a team of up to two people.

- All team members are responsible for the project files submission
- No collaboration, cheating, and plagiarism is allowed in assignments, quizzes, the midterms or final project.
- All the submissions will be checked using SafeAssign as well as other tools. SafeAssign is based on the submitted works for the past semesters as well the current submissions. We can see all the sources, so you don't need to worry if there is a high similarity with your Checkpoint submission.
- The submissions should include all the references. Kindly note that referencing the source does not mean you can copy/paste it fully and submit as your original work. Updating the hyperparameters or modifying the existing code is a subject to plagiarism. Your work has to be original. If you have any doubts, send a private post on piazza to confirm.
- All group members and parties involved in any suspicious cases will be officially reported using the Academic Dishonesty Report form. What does that mean?
    - In most cases, the grade for the assignment/quiz/final project/midterm will be 0 and all bonus points will be subject to removal from the final evaluation for all students involved.
    - Those found violating academic integrity more than once throughout their program will receive an immediate F in the course.

  Please refer to the Academic Integrity Policy for more details.
- The report should be delivered as a separate pdf file. You can combine report for Part I, Part II & Part III into the same pdf file. You can follow the NIPS template as a report structure. You may include comments in the Jupyter Notebook; however, you will need to duplicate the results in a separate pdf file.
- All the references can be listed at the end of the report. There is no minimum requirement for the report size, just make sure it includes all the information required.

- For the Bonus part, no report is needed.

Late Days Policy

You can use up to 7 late days throughout the course that can be applied to any assignment-related due dates. You do not have to inform the instructor, as the late submission will be tracked in UBlearns.

If you work in teams, the late days used will be subtracted from both partners. In other words, you have 4 late days, and your partner has 3 late days left. If you submit one day after the due date, you will have 3 days and your partner will have 2 late days left.

Important Dates

February 17, Friday - Register your team (UBLearns > Tools > Groups)

February 23, Thursday - Checkpoint is Due

March 9, Thursday - Final Submission is Due