AI ASSIGNMENT 4
Date of Submission : As on GC

Total Marks : 25 Marks
Linear Regression Gradient Descent: 10 Marks
Linear Regression (OR) Logistic Regression : 15 marks

Instructions:

1. Assignments are to be attempted individually.
2. Submit the assignment in a zipped file with name AI4_⟨StudentName⟩ and contents including AI4_StudentName_Result.pdf and ⟨StudentName⟩_gradient.py and the computational question as ⟨StudentName⟩_linear_regression.py, and ⟨StudentName⟩_logistic_regression.py depending on which question you solve.
3. STDOUT / python print outputs will be considered for evaluation.
4. Plagiarism checking will result in deducting of marks but doesn't count for overall marking scheme.
5. **Programming Language :** Python
6. Use classes, functions etc. in your code in ways that make sense.
7. **Extension and Penalty clause:**
   - A penalty will be incurred if the submissions are late.
   - Please make sure to turn in the Assignment. Missing files will receive zero marks.

1. (a) (5 marks) Derive the gradients of linear regression with 5 variables using expanded equation form and vector form and write the gradient descent update rule for the weights using Mean Squared Error (MSE) as loss function. Typically, gradient descent is run on the entire dataset (full gradient descent) or mini-batches (batched gradient descent). This problem is simple enough that you should be able to get results using per example iterations and MSE for every y_p individually without summing it over mini-batches or the full dataset.

   Expanded equation form:

$$y = w_1 * x_1 + w_2 * x_2 + w_3 * x_3 + w_4 * x_4 + w_5 * x_5 + b$$

   Vector form:

$$\mathbf{w} = [w_1, w_2, w_3, w_4, w_5]^\top, \mathbf{x} = [x_1, x_2, x_3, x_4, x_5]^\top, b$$
$$y = \mathbf{w}^\top \mathbf{x} + b$$

Here, $\mathbf{w}$ are the weights and b is the offset.
Mean Squared Error:

$$\begin{aligned}\mathcal{L}(\mathbf{w}; \{(\mathbf{x}^(i), y^{(i)})\}) &= \frac{1}{N} \sum_{i=1}^{N} \left(y_p^{(i)} - y^{(i)}\right)^2, \qquad i = 1, 2, \ldots, N \\ &= \frac{1}{N} \sum_{i=1}^{N} \left(\mathbf{w}^\top \mathbf{x}^{(i)} - y^{(i)}\right)^2, \qquad i = 1, 2, \ldots, N\end{aligned}$$

Here $\{(\mathbf{x}^{(i)}, y^{(i)}); i = 1, 2, \ldots, N\}$ are the labeled data samples available for the optimization. $y_p^{(i)}$ are the predictions from the linear regression model.

Submit the derivation of the gradients for the above 5-dimensional example, both in the expanded equation form as well as the vector form.

(b) (5 marks) Implement the gradient descent for linear regression as a loop over 100 iterations where you alter the weights manually using a small enough learning rate instead of using scikit-learn. Use the following (x, y) pairs and learn w.

```
X = np.arange(-20, 20, 0.1)
np.random.shuffle(X)

eps = np.random.rand(400) * 10
y = 23*X + 43 + eps
```

EXAMPLE OUTPUT STDOUT: w = 23, b = 47

Please use the same notation in your python code's print statement. Evaluation will take running time into account.

Environment Setup: pip install ucimlrepo pandas scikit-learn
By popular demand, it has been decided that only one of the following questions needs to be answered to complete the assignment. Please choose either Linear regression or Logistic regression and submit the filenames accordingly. Thanks.

2. Linear Regression (15 marks)

(a) **Reference:** Regression using continuous variables can be applied directly. Categorical variables, on the other hand, need some processing before they are used in regression. Look into the different approaches used here and implement your choice(s) by analyzing the data: https://stats.oarc.ucla.edu/spss/faq/coding-systems-for-categorical-variables-in-regression-analysis-2/. Typically, one-hot encoding gives independence between the regression weights for the categories and assumes an independence between the categories that might not be useful and hence there are other ways people have come up with.

(b) **Dataset:** Abalone (https://archive.ics.uci.edu/dataset/1/abalone) - use python to download the data instead of the web interface.

(c) **Accuracy:** R^2 score

(d) **Target variable:** Rings (integer)

(e) **Target accuracy (10 marks):** 0.56 over entire dataset

(f) **Cross Validation (5 marks):** Take subsets of the dataset using train val and test splits of 70%, 15% and 15% and find the average test R^2 score over different randomized splits of the data by using validation set as the deciding factor for performance. Run around 15-20 sets for the average performance and standard deviation on R^2 metric.
Look into this for some context on the r^2 metric:
scikit-learn documentation:https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_s
Examples: https://www.ncl.ac.uk/webtemplate/ask-assets/external/maths-resources/statistics/r
and-correlation/coefficient-of-determination-r-squared.html.

(g) **Submission Format:** ⟨StudentName⟩_linear_regression.py

- R^2 score over entire dataset: R^2 =0.537 minimum; R^2 =0.56+
- R^2 score over 15% test dataset: mean R^2 score and standard deviation over 15% test dataset
- The assessment will be automatic. Please use the EXACT STRINGS as shown in the example output
- The only outputs needed are the performance metric numbers changed according to your results.

(h) **Example Output STDOUT:**
Full dataset train and eval R2 score: 0.54
70-15-15 Cross validation boxplot: mean=0.51, std=0.04

(i) Visualize the box-plots on matplotlib for different dataset splits to understand regression more if you can. No marks are given for this.

(j) **HINT:** You can try to use non-linear transformations of the data for performance ablations.

3. Logistic Regression (15 marks)

(a) **Reference:** This question contains data with missing values. Here is a reference you can use for common strategies employed in regression datasets with missing values: http://www.stat.columbia.edu/ gelman/arm/missing.pdf. Consider using matching, mean value + random noise or more feature-specific random imputations of a single variable in your free time. For now, you can assign an UNKNOWN label to all missing values and try to approximate the results anyway. Note: Logistic regression uses LBFGS in the backend.

(b) **Dataset:** Adult (https://archive.ics.uci.edu/dataset/2/adult) - use python to download the data instead of the web interface.

(c) **Accuracy:** Classification score

(d) **Target variable:** income (binary)

(e) **Target accuracy (10 marks):** 0.80 over entire dataset

(f) **Cross Validation (5 marks):** Take subsets of the dataset using train val and test splits of 70%, 15% and 15% and find the average test accuracy score over different randomized splits of the data by using validation set as the deciding factor for performance. Run around 15-20 sets for the average performance and standard deviation on accuracy metric.

(g) **Submission Format:** ⟨StudentName⟩_logistic_regression.py

- accuracy score over entire dataset: full accuracy: 0.80 ; train accuracy=0.79 ; test accuracy=0.78
- accuracy score over 15% test dataset: mean accuracy score and standard deviation over 15% test dataset
- The assessment will be automatic. Please use the EXACT STRINGS as shown in the example output
- The only outputs needed are the performance metric numbers changed according to your results.

(h) **Example Output STDOUT:**
Full dataset accuracy: full: 0.80, train: 0.79, test: 0.78
70-15-15 Cross validation boxplot: mean=0.74, std=0.08

(i) Visualize the box-plots on matplotlib for different dataset splits to understand regression more if you can. No marks are given for this.

END