IIT BOMBAY

# Lab Assignment #3
# Points: 5

Main TAs: *P S V N Bhavani Shankar, Dhiraj Sah, Vedang Dhirendra Asgaonkar*

Course: *CS 335* – Instructor: *Preethi Jyothi*
Due date: *August 21, 2023*

**General Instructions**

- Download lab3.tgz from Moodle for CS 335, and extract the file to get `lab3`. The data files are also up on Kaggle's "Data" tab (`https://www.kaggle.com/t/3780e6d86cad4ca7b85351b538a7bea4`).
- The folder `splits/` contains all the CSV files required for this task. Make sure you update the path files in the template code to load the CSV files properly.
- `lab3_template.py` has template code for all three questions. `Two_Way_Interactions.pdf` contains a detailed example that will help with Q2.
- For your final submission, submit the source file with the TODOs filled in and name it as `[rollnumber]_template.py`. This lab submission is due on Moodle on or before **11.59 pm on Aug 21, 2023**.
- You will get 1 point if your regularized solution improves over the unregularized solution in terms of test RMSE. You will get 3 points if your code for two-way interaction passes our test cases. You will get 1 point if your ElasticNet solution match es or improves over the regularized solution. Your roll number **HAS TO** appear on the Kaggle leaderboard to get full points for this lab assignment. You can get up to 3 extra credit points if you top the Kaggle private leaderboard. More details appear at the end.

## Regularized Linear Regression

### Q1: L2 and L1-Regularized Regression

This task involves integrating **L1 and L2 regularization** techniques into the existing gradient descent code, provided in the template file `lab3_template.py` file. Regularization is pivotal in mitigating over-fitting by introducing penalty terms to the loss function and discouraging very large parameter values. L1-regularization enforces sparsity by driving some weights to zero, whereas L2-regularization shrinks all weights towards zero. The aim is to optimize model performance by striking a balance between minimizing the loss and controlling the magnitude of the parameters. An added advantage is that regularization can effectively prevent over-fitting even with reduced training data, as it constrains the model's complexity and curbs its tendency to fit noise. You will explore this further in Q1.

The code for loading datasets, computing batches, evaluating on the `test_set`, and saving the predictions on `hidden_test_data` are already implemented.

Complete the following functions:

1. `compute_gradient()`: Complete the function to compute gradients of the regularized mean-squared error loss with respect to the weights. Add `if-else` statements to differentiate between L1 and L2 regularization penalty terms.

2. `grid_search()`: For each regularizer (L1, L2), implement a grid search over the regularization strengths and compute the losses on the development set. This function returns `dev_losses`, `best_dev_loss` and `best_params` which are the RMSE losses on the dev set across all hyperparameters in the grid search, the minimum dev RMSE loss and the hyperparameters that correspond to this minimum loss, respectively.

## Q2: Two-way Interaction-based Features

Basis functions aim at deriving new features from existing ones to capture more intricate (non-linear) relationships within the data. A two-way interaction between two features, $A$ and $B$, can be represented as the product of their values, i.e., $A \times B$. These interactions can sometimes unveil patterns in the data that are not captured by individual features.

For this task, you will be developing a function to compute unique two-way interactions for a dataset using numpy and vectorized operations. Your function `get_two_way_interactions` should be able to:

- Select one-third of the features from the input dataset. (This is to ensure that the resulting expanded feature set is sufficiently small and runs in reasonable time. You can choose to expand this to the complete feature set for the extra credit part.)
- Efficiently compute unique two-way interactions for these features. Ensure that duplicate interactions (due to the commutative property of multiplication) are avoided.
- Concatenate these interaction features to the original feature matrix.
- Use vectorized operations (numpy) and avoid using for loops.

**Hints:**

- Consider using numpy's broadcasting capabilities to compute interactions without resorting to nested loops.
- To avoid duplicates, think of how you can create a mask for the upper triangle of a matrix.
- A simple example is detailed in *Two_Way_Interactions.pdf*

The goal is to understand a) how to compute two-way interaction features using vectorized operations, and b) how adding interaction-based features can influence model performance.

## Q3: Elastic Net

Often machine learning tasks involve datasets where features are correlated with each other. Regularizing such datasets using the L1 regularizer induces sparsity and prevents overfitting, but often only one of the correlated features is retained by the L1 regularizer thus hampering model performance. In such settings, one could soften the sparsity inducing tendency of the L1 regularizer via a linear combination with the L2 regularizer. This is known as an *Elastic Net* regularizer, computed as:

$$\lambda(\alpha||w||_1 + (1-\alpha)||w||_2^2)$$

where $\lambda$ controls the amount of overall regularization and $\alpha$ adjusts for sparsity induction.

For this part, you will modify the following functions:

- `compute_gradient()`: Implement the gradient term that will get added to the weights due to ElasticNet regularization.
- `grid_search_ElasticNet()`: Implement a grid search by tuning the hyperparameters $\lambda$ and $\alpha$.

Note that for grid search, you may implement a full 2-dimensional grid search or opt for a heuristic-based variant where you begin with a reasonable $\alpha$ and optimize $\lambda$, followed by fixing $\lambda$ and optimizing $\alpha$. The heuristic employed here is that $\lambda$ impacts the performance of the model on a larger scale than $\alpha$. Note that the grid search for $\lambda$ should be done on a geometric scale $(0.001, 0.01, 0.1 \cdots)$, while that for $\alpha$ should be done on a linear scale $(0.1, 0.2 \cdots)$

## ❊ Extra Credit: Climb the Leaderboard on Kaggle

The objective for this extra credit part is to build the best possible regularized linear regression model using any enhancements you like. Submit your target predictions for the instances in `hidden_test_data.csv` to Kaggle at `https://www.kaggle.com/t/3780e6d86cad4ca7b85351b538a7bea4` so that your roll number appears on both the "Public Leaderboard" (and eventually the "Private Leaderboard" after the assignment concludes). Top-scoring performers on the "Private Leaderboard" (with a suitable threshold determined after the deadline passes) will be awarded up to 3 extra credit points. The exact breakdown of the three points for this question will be announced later.