TP : Gradient descent for linear models

1 The 1 dimensional case

1. **Implement Gradient Descent Function**
   Code a Python function for gradient descent. The function should receive the feature matrix X and target vector y, along with the learning rate and the number of iterations. As a first step in the algorithm, include a column of ones in the feature matrix X to account for the intercept.
2. **Create a Random Sample Dataset**
   Utilize the `make_regression` function from `sklearn.datasets` to create a random dataset with the following characteristics :
   - 1 independent variable.
   - 100 samples.
   - Noise set to 30
   - An intercept of 10.
   - Random seed set to 0.

   Plot the generated dataset, and take note of the slope used for its generation by referring to the function's manual.
3. **Apply Gradient Descent**
   Call the gradient descent function implemented in Step 1 using the generated dataset. Plot the following :
   - The evolution of the Mean Squared Error (MSE) with respect to the number of iterations.
   - The evolution of the L2 norm of the gradients with respect to the number of iterations.
   - Plot the data points and all the solution lines at each step of the gradient descent algorithm in the same plot. Bonus : use `plt.cm.bwr` to get a colormap that transitions from yellow to black so that the colors of the lines change for each iteration of the algorithm.
4. **Compare Different Estimation Methods**
   Compare all the alternative methods for estimating the linear regression model :
   - The real slope and intercept used for dataset generation.
   - The solution obtained using the normal equation.
   - Te solution of `sklearn`.
   - The solution obtained using the gradient descent algorithm implemented in Step 1.

   Analyze and discuss the differences and similarities among these estimation methods.

2 Exploring the `diabetes` Dataset

In this section, we will work with the `diabetes` dataset to perform various tasks related to data preprocessing and gradient descent optimization.

1) **Data Loading and Separation**
   Load the `diabetes` dataset. Split the data into the design matrix X (features) and the observed values y (target variable).
2) **Train-Test Split**
   Use the `train_test_split` function to split the data into training and test sets. Allocate 33
3) **Standardization of Features and Observations**
   Standardize both the features and observations using the `fit_transform` and `transform` functions. Ensure that you use both transformations to prevent data leakage.

4) **Performance Metrics and Visualization**
Perform the following visualizations and calculations :

- Plot the Mean Squared Error (MSE) as it evolves with the number of iterations.
- Plot the L2 norm of the gradients as they change with each iteration.
- Create a heatmap displaying the evolution of gradients for each coefficient (in rows) across different iterations (in columns).

These exercises will help you gain practical experience in handling datasets, preprocessing data, and visualizing optimization processes using the `diabetes` dataset.