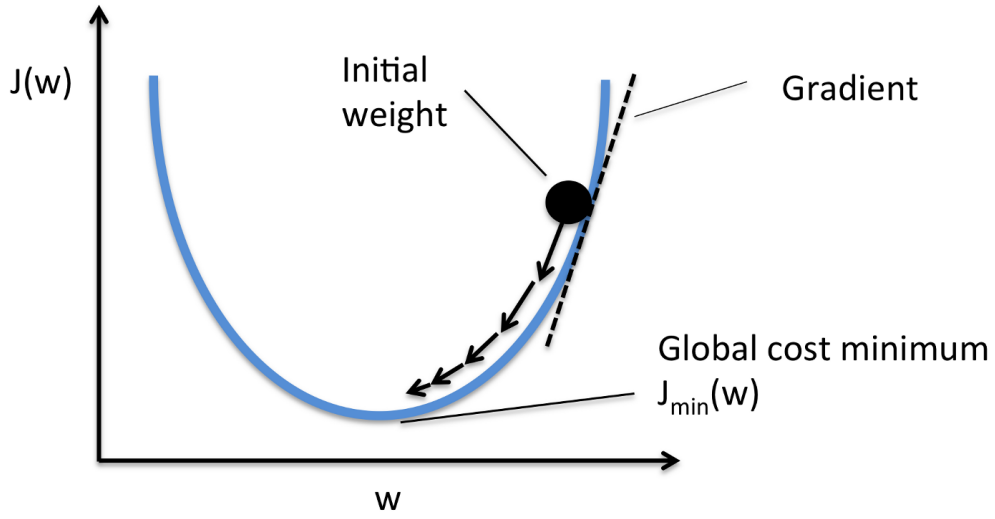
CS 335: Lab Assignment 5

(TAs in charge: Rishabh Shah, Vishwajeet Singh)

In this assignment, you will perform Ridge and Lasso Regression using the old school method of gradient descent.



Standard Gradient Descent Algorithm
(Image source: https://cdn-images-1.medium.com/max/800/1*HrFZV7pKPcc5dzLaWvngtQ.png)

Data Set

You are given the standard House Price Prediction Dataset. With 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa, the challenge is to predict the final price of each home. There are 1461 number of data points available that have been segregated to training data containing 1200 points and test data containing 261 points.

One part of this assignment entails coding up the gradient descent algorithm from scratch and applying it to ridge regression. In second part, we learn a new type of gradient descent to be applied to Lasso regression.

House Pricing Dataset
(Image source: https://storage.googleapis.com/kaggle-competitions/kaggle/5407/media/housesbanner.png)

Code

The base code for this assignment is available in this compressed file. Below is the list of files present in the `Lab5_base` directory.

File Name	Description
`tasks.py`	This file contains all the required functions that you need to complete.
`utils.py`	This file contains all helper functions that you can use.
`autograder.py`	This is used for testing your tasks 1,2,3,4.

The data set is provided in the `dataset` directory.

Methods you need to implement

You will only have to write code inside the following functions.

- `preprocess`: This method takes as input the raw csv extracted data (contains strings, numbers, floats...) output from `read_data` function and returns the processed data (containing all numerical values). For categorical string data you should use the `one_hot_encode` function and for other numerical data use following normalisation.

$$(X_{new_i}) = \frac{X_{old_i} - \mu(X_{old})}{\sigma(X_{old})}$$

NOTE: Do not use normalisation for output values

- `ridge_grad_descent`: This method takes as input training data `X` and with labels `Y` along with hyper parameters such as learning rate, max_iterations, λ, and returns the trained weight vector in form of a numpy array. You can initialize the weights in any way. You have to call function grad_ridge to get the error gradient and update the weight vector according to the ridge loss function.

$$R(w) = \sum_{i=1}^{n} \|y_i - x_i.w\|_2^2 + \lambda \|w\|_2^2$$

  where `R(w) is the ridge objective function w.r.t. w` and
  `w is the weight vector` and
  `x_i is i_th data input vector` and
  `y_i is i_th output vector` and
  `n is total number of input points`.

  NOTE: Remember to follow same convention for matrix dimensions as specified in the comments of the task.py file. Also, you may create global variables and precompute some values to increase the speed of your algorithm.

- `grad_ridge`: This method takes as input the train input `X`, train output `Y`, λ and weight vector `W` and returns the $\frac{dR(w)}{dw}$.

- `k_fold_cross_validation`: This method takes as input train data `X` and train labels `Y`, number of splits k along with a list of lambdas and algorithm on which we want to tune hyperparameter lambda to get best possible test SSE. In this function, you will first split the data into k equal parts, perform cross validation to get SSE score (on validation data) for each of the lambda. Plot this score w.r.t. lambdas and choose the lambda that minimizes the above score
  NOTE: For purpose of this lab, while performing k fold cross validation, do not reshuffle the data before making splits.

- `coord_grad_descent`: This method takes as input train data `X` and train output `Y` along with hyper parameters like max_iterations, lambda and returns the trained weight vector in form of a numpy array optimizing the Lasso loss function.

$$L(w) = \sum_{i=1}^{n} \|y_i - x_i.w\|_2^2 + \lambda \|w\|_1^1$$

  where `L(w) is the lasso objective function w.r.t. w`.

Since the l_1 -regularization term in the objective function is non-differentiable, it is not immediately clear how gradient descent or SGD could be used to solve this optimisation problem directly. One approach to solve such problems is coordinate descent in which at each step we optimise over one component of the unknown parameter vector, fixing all other components. The descent path so obtained is a sequence of steps, each of which is parallel to a coordinate axis. It turns out that for the Lasso optimization problem, we can find a closed form solution for optimisation over a single component fixing all other components. This means that while updating the kth component of the weight vector, solving for $\frac{dL(w)}{dw_k} = 0$ will give the new w_k value.

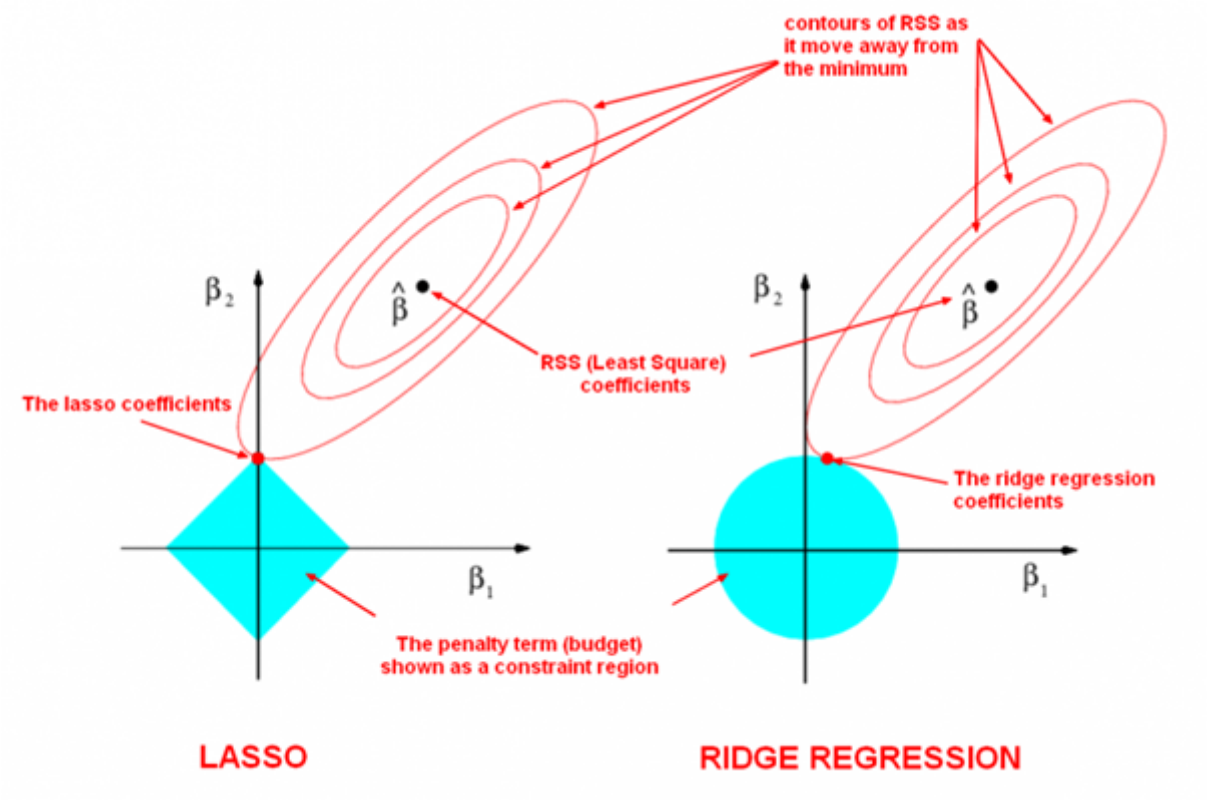
This gives us the following algorithm known as Coordinate Gradient Descent:

- Initialize weight vector w.
- Repeat until converged/max_iterations :
  - For every dimension of input:
    - Set w_k = w such that $\frac{dL(w)}{dw_k} = 0$.

NOTE: Technically, you would need to use the subgradient of $L(w)$ as it is non differentiable to solve $\frac{dL(w)}{dw_k} = 0$

Also, solve this equation manually (not by gradient descent), as closed form solution is much faster and easier to compute.

Refer to this link for further clarifications on Coordinate Descent Algorithm



Differences between Ridge and Lasso regression
(Image source: https://qph.fs.quoracdn.net/main-qimg-2a88e2acc009fa4de3edeb51e683ca02.)

Task 1: Preprocessing (1 Mark)

In this task, you will complete the `preprocess` function in `task.py` file. To test your code, run the following command.

```
python3 autograder.py 1
```

Task 2: Ridge regression (2 Marks)

In this task, you will complete the `ridge_grad_descent` and `grad_ridge` function in `task.py` file. You are free to change the default parameters like learning rate, epsilon, max_iter as you wish (as long as it passes autograder). To test your code,

run the following command.

```
python3 autograder.py 2
```

Task 3: K Fold Cross validation (1 + 1.5 Marks)

In this task, you will complete the `k_fold_cross_validation` function in `task.py` file. To test your code, run the following command.

```
python3 autograder.py 3
```

Note: This is just sanity check test (1 Mark). You will get full marks for tuning the lambda appropriately for both ridge and lasso regression. Report the hyperparameters used in observations.txt. Also, report the final SSE obtained on Test data (261 points). Make use of `plot_kfold` function to plot the SSE v/s lambda plot for ridge and lasso parts. Save them as `ridge_kfoldcv.png` and `lasso_kfoldcv.png`. Also, explain in observations.txt how does this plot help you tune the lambda

---

Task 4: Lasso Regression (2.5 Marks)

In this task, you will complete the `coord_grad_descent` function in `task.py` file. Make sure to write efficient code for this task (i.e 2000 iterations should take less than a min) for doing a fast cross validation. You are free to change the default parameter max_iter as you wish (as long as it passes autograder). To test your code, run the following command.

```
python3 autograder.py 4
```

Task 5: Observations (2 Marks)

In this task, answer the following question.
Is there something unusual with the solution of Lasso compared to the Ridge? Explain why such a thing would happen? Is using lasso advantageous compared to ridge. How?

Submission

You are expected to work on this assignment by yourself. You may not consult with your classmates or anybody else about their solutions. You are also not to look at solutions to this assignment or related ones on the Internet. You are allowed to use resources on the Internet for programming (say to understand a particular command or a data structure), and also to understand concepts (so a Wikipedia page or someone's lecture notes or a textbook can certainly be consulted). However, you **must** list every resource you have consulted or used in a file named `references.txt`, explaining exactly how the resource was used. Failure to list all your sources will be considered an academic violation.

In `task.py`, you will have to fill out functions for all the tasks specified.
In `observations.txt`, you will report the parameters (learning rate, max_iteration, lambda used) for both lasso and ridge that was used to obtain your min SSE on test data. Also, explain the plots and make sure to answer the Task 5 completely here.
Also, submit the kfold cross validation plots `ridge_kfoldcv.png` and `lasso_kfoldcv.png`.

Remember to test your solution before submission using the autograder provided.

```
python3 autograder.py [task_num]
```

where `task_num` can be one of {1,2,3,4, all}.

Place these four files (and additionally `references.txt`) in a directory (la-5-rollno), and compress it to be la5-rollno.tar.gz where rollno is your roll number (say 12345678, then Final compressed file would be la5-12345678.tar.gz) and upload it on Moodle under Lab Assignment 5.