Report Project 3

Emanuele Falci, Simone Varriale

19/01/2024

1 Part I - Implementation ridge regression

In this lab we implement the ridge regression, a method used to regolarize the linear regression. This can be of use when the model can not generalize well for example if the features are strongly correlated.

Our class will have the following fields:

- **X**, **y**, **weigths**: not different from other models, they are respectively the training set, the true label of the trainig set, the importance of each feature (updated during the training phase).
- **λ**: the strength of the regularization, how much we are going to regularize our weights inside the objective function. This is an hyperparameter for which we need to perform validation.
- **normalized**: flag to opt in the normalization of the data.

After having defined the base structure of our class we proceed with the code for the training and prediction:

- ***train(self, X, y)***: X and y are the training set and true label of the sample inside the trainig set. If the normalized flag is set, the data is standardized in order to make the model more stable (see function below). We add a column of 1 to X to be able to embed the offset w0 in w. In order to obtain the weigth we use:

  $$w = (X^T X - \lambda * I)X^T y$$

  Where I is an identity matrix with dimension equal to the number of samples in the training set and the first element (0,0) set to 0 because the offset does not have to be regularized.
- ***predict(self, X_test)***: the predict method compute the prediciton with the following formula:

  $$y = self.X * self.weigths$$

- ***embed_bias(self, X)***: our version of PolynomialFeature from scikit learn, it adds a first column of 1 to the X matrix.
- ***MSE(self, y_hat, y)***: to meausure the performance of the model.
- **normalize(self, X, y)**: in order to make the model more stable we apply z-score normalization on the features. It is useful in dataset with more feature in order to avoid the predominance of one of them. Centering and normalizing the data reduce the relevance of the effect of the intercept. The bias is not standardized since it is not affected by the scaling of the rest of the features.

2 Part II - Using the ridge regression

Once the implementation is ready we try to generate a model for the *Olympics 100* dataset and the code is provided in the *experiments.py* file.

The newelty of this lab is having to tune the hyperparameter λ through a process of validation. Since the cardinality of the dataset is very small (only 29 samples), we decided to use *k-fold cross validation*. Unfortunately, we can not use the usual K-fold CV algorithm because the dataset is a time series, that means that we must pay attention not to use data from the future to predict data from the past. To address this problem we can make use of a variant of validation, considering the number of split this validation will be repeated k times covering all dataset going forward in time. The algorithm splits the dataset into consecutive folds, where each fold adds data from a later time than the previous fold, and the test data is taken from the latest data in the fold.

The following steps were implemented both with and without normalizing the data:

- **Split** of the dataset in 80%-20% (training and test), without shuffling since we are in a time serie.
- **CV**: The training set is further divided through with the SplitTimeSeries function from scikit. It enabled us to get the folds to perform the CV (with k=5 typical number of folds).
  - Define a range of λ that goes from 10^{-10} to 10^3.
  - Train the model using k-fold leaving λ fixed. (In a second run with normalization option).
  - For each lambda, compute the average of the MSE resulting in the diffent k-folds.
  - Choose the λ that gave us the minimum error on average.
  - Repeat the same procedure a second time to find a finer value of lambda.
- **Re-Train**: once the value for λ is set, we retrain the model over the full training set to have the final model.
- **Test**: we then test the model on the original test data, and calculate the relative MSE.
- **Comparison** with scikit and graph: in the majority of the case our model and the sk-learn's model performances (in terms of MSE value) are basically the same. Some changes can be found at the 12th decimal value. We plotted the 2 linear model resulting from the two models (Figure 1).

DISCLAIMER: We used ChatGPT to generate a first draft of the documentation for the class.

APPENDIX:

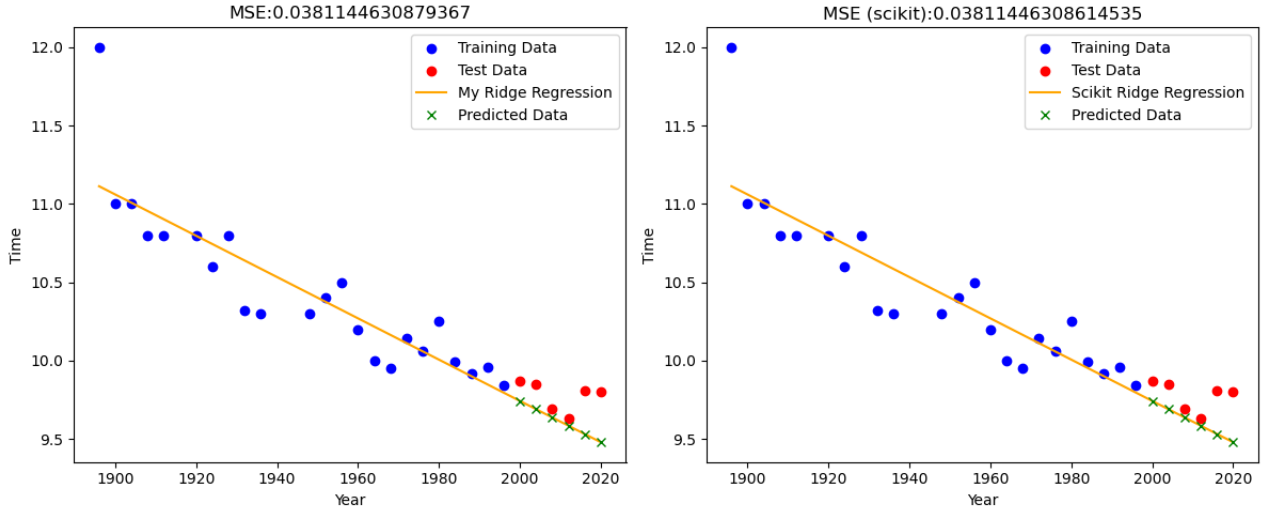

Figure 1: Plot of our model (L) and sk-learn model (R) without normalization

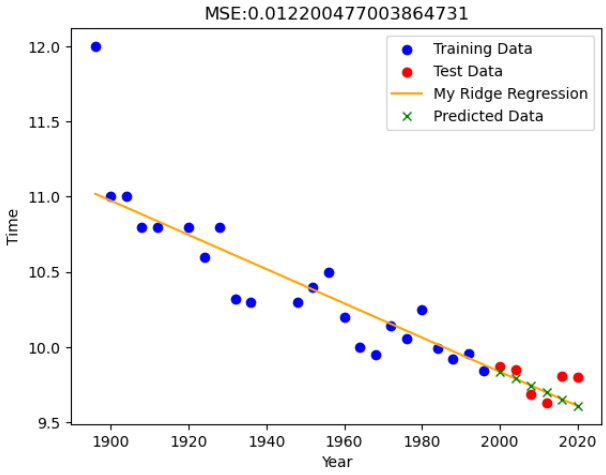

Figure 2: Plot of our model with normalization of data