1 Implementation of Ridge Regression

Ridge Regression is a linear model for regression that takes into account a regularization term in the objective function. Starting from the linear regression objective function:

$$argmin_w \Lambda(w) = argmin_w \frac{1}{N}(y - Xw)^T(y - Xw)$$

We include the L2-regularization term[1]

$$argmin_w \Lambda(w) + \lambda R(w) = argmin_w (y - Xw)^T(y - Xw) + \lambda w^T w$$

Thus obtaining the closed form solution for w (**Ridge regressor estimate**)

$$\hat{w} = (X^T X + \lambda I)^{-1} X^T y$$

In this way, we are able to keep the values of w low, since very large values of w could make our model very sensitive, thus leading to poor generalization. Furthermore, constraints added via regularization may help us in solving ill-posed problems, when $X^T X$ is not possible to invert (when $D >> N$). Therefore, λ becomes an hyper-parameter of our model controlling the strength of the regularization. After obtaining the vector of weights $\hat{w}$ we can perform regression in the same way as a non regularized linear regression in the form: $y = \hat{w}^T x$

1.1 The train function

For the training phase we simply need to compute the closed form solution to obtain our weights. We only need to pay attention in transforming our data in order to absorb the bias term and set $I(0, 0) = 0$ before computing the solution, so that we do not regularize the bias term w_0

1.2 The predict function

Again, the predict function is very simple, we just need to transform the data we want to predict in order to take into consideration the bias term, and then compute[2] $Y = Xw$

2 Validation of the model

Since we are working on a dataset (Olympics 100m) with very few samples (29), after splitting it in training set (80%) and test set(20%), we performed **cross validation** with a **coarse-to-fine** approach to choose the best value of λ. However, since we are in the case of time series, cross validation and train/test splitting are not trivial, we cannot choose random samples and assign them to either the test set or the train because it makes no sense to use values from the future to forecast values in the past, we need to preserve the temporal dependency between observations. Thus we performed the first split in train/test taking into account this issue (by not shuffling). Then, to perform validation, for each value of λ, we start with a small subset of data for training (in our case we chose half of our training set), predict the temporally next sample, and compute MSE on this prediction[3]. Then we include the same predicted sample as part of the next training dataset and subsequent point is forecasted, and so on (see figure 1a for a general idea). Then we average the MSEs obtained for these to obtain a an average value of MSE that we use to evaluate the goodness of the given λ. Doing so, we manage to do a cross validation consistent with time series and we obtain that the best model is with $\lambda = 2548.65$, with $MSE = 0.0217$.

3 Test of the model

We also tested our best configuration on a never-seen test set (20% of our dataset) after training on our full training set, and we obtained an $MSE = 0.0229$. We can say that our model generalized well.

4 Comparison with scikit-learn

We then also compared our implementation of the Ridge Regression with the one of scikit, using the λ we selected as the best one. As we can see in the plot, the two are equal.

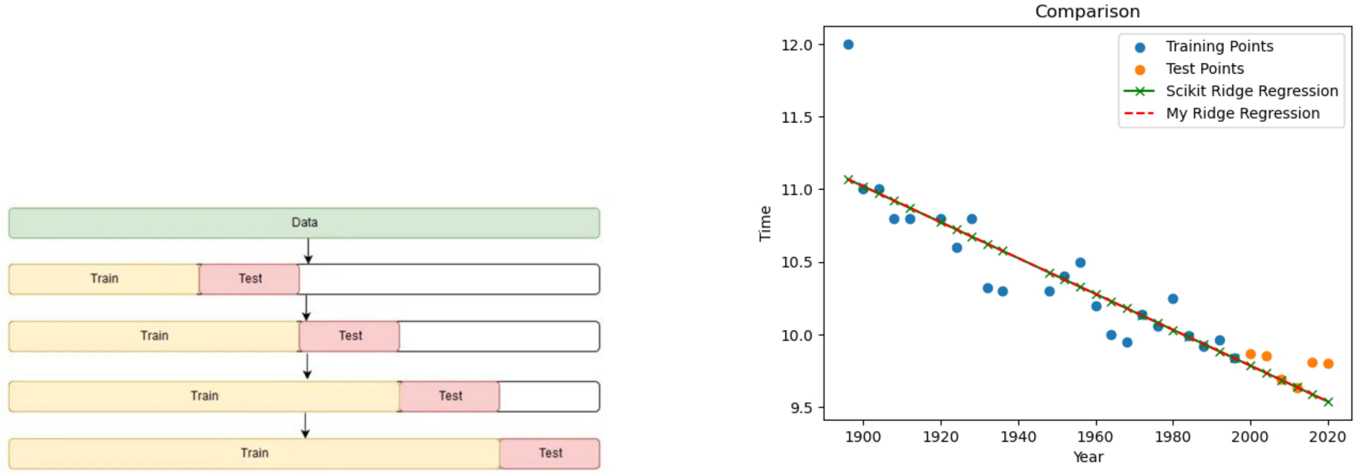
- Our implementation: $w_0 = 34.465, w_1 = -0.0123, mse_{test} = 0.0229$
- Scikit: $w_0 = 34.465, w_1 = -0.0123, mse_{test} = 0.0229$

The results are different only starting from the 10th decimal.

[1] Ridge Regression uses L2-regularization term

[2] We are doing matrix multiplications, thus we obtain an array of predicted values

[3] source for this method: https://medium.com/@soumyachess1496/cross-validation-in-time-series-566ae4981ce4



(a) Cross validation for time series

(b) Comparison with SciKit

Figure 1

5 ChatGPT Policy

We used ChatGPT to generate a first draft of our code documentation for the Ridge class.