Regression and forecasting

Project description

The aim of this project is to perform regression and forecasting on a simple toy dataset and on a more complex real dataset. In particular, the goal is to assess the performance of different linear regression models: Ordinary Least Squares (OLS), Lasso and Ridge, and a non-linear regression model using neural networks.

The regression task can be generally described as a way to estimate the relationship between a dependent variable y and an independent variable x. This relationship can be formulated as:

$$y = f(x) + \epsilon \tag{1}$$

where $f(x)$ represents the underlying true dependency between x and y, and ϵ is a measure of the noise introduced in the process of measuring y.

The difference between regression and forecasting is that in the latter the independent variable is usually time, and the goal is to predict what will happen in the future given the present and past data. Forecasting is essentially a process of extrapolation in time, which can become very complicated if the dataset comes from a chaotic system (like weather forecasting). However, often it is possible to exploit the quasi-periodicity of the dataset to improve the accuracy.

In selecting the polynomial regression, we are restricting the function $f(x)$ to assume a polynomial representation:

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_p x^p \tag{2}$$

The training of the regression model will correspond to the selection of the β_j parameters, such that the model's prediction 'fits' the observed values.

The selection of the optimal parameters is equivalent to the solution of the linear system of the type $\mathbf{Ax} = \mathbf{b}$:

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^p \\ 1 & x_2 & x_2^2 & \dots & x_2^p \\ \vdots & \vdots & \ddots & \vdots & \\ 1 & x_n & x_n^2 & \dots & x_n^p \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{3}$$

The linear system admits a single solution only if the matrix $\mathbf{A}$ is invertible. This requires that $n = p$ and that $\mathbf{A}$ is a full rank matrix. In most cases, this requirement is not fullfilled and the system can admit no solutions (overdetermined system) or infinite solutions (undertermined system). In this case, the best combination of parameters β_i is found following an optimization process, where the objective is to minimize a loss function. The difference between OLS, Lasso and Ridge lies in the formulation of the loss function.

Contrary to the polynomial regression, neural networks don't make any assumption on the form of the function $f(x)$, and in principle they can represent any arbitrary function, given enough nodes and layers.

Tasks

In this project, you will complete the following tasks:

Task 1: Simulate the trajectory of a projectile with initial velocity magnitude $v_0 = 10\ \mathrm{m/s}$ and initial position $x_0 = 0\ \mathrm{m}, y_0 = 2\ \mathrm{m}$ for the time it takes to reach the ground. The launch angle θ is equal to $50°$. The x, y position as well as the velocity magnitude $v = \sqrt{v_x^2 + v_y^2}$ are tracked by a sensor which has a $1\ \mathrm{m/s}$ uncertainty in the velocity measurement and

a 0.5 m uncertainty in the position measurement. This is equivalent of saying that the measured value is sampled from a normal distribution with mean equal to the true value and standard deviation equal to the experimental uncertainty. The position and velocity is tracked every 0.01 s. The sketch of the synthetic experiment is reported in Fig. 1.

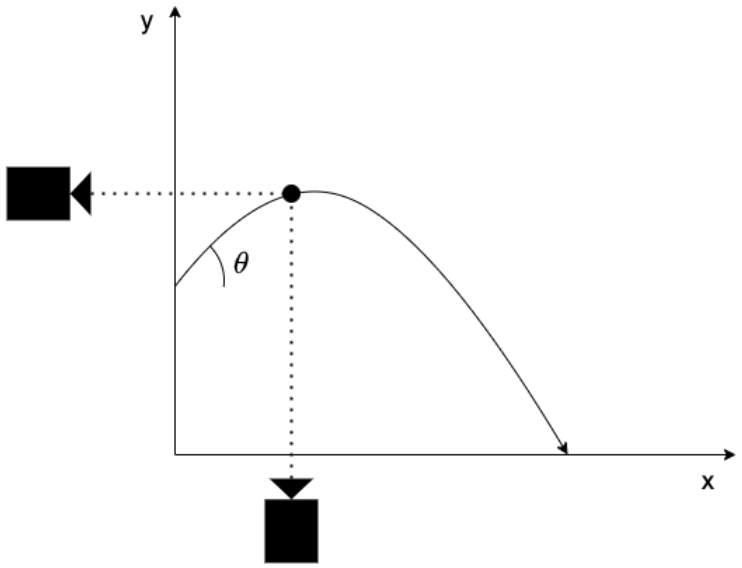

Figure 1: Sketch of the virtual experiment.

The steps to solve the first exercise are:

- Build a 2 degree polynomial regression using the OLS method to predict the y position of the projectile using the noisy x measurements as input.
- Build the same OLS regression model using a 20 degree polynomial. Then compare the obtained prediction with the Lasso regression model. Try different values of the `alpha` parameter, and use the cross-validation to find the best value.
- Repeat the same process by applying the `Ridge` and `RidgeCV` regressions and compare the results to the OLS results.
- Compare the Lasso and Ridge regression results in extrapolation. To do that, you can predict y by using as input a vector x which is 30% bigger than the one used for training.
- Build a simple fully-connected neural network to perform the regression and compare it to the results you obtained using polynomial regression.

Task 2: Build a regression model to predict the daily energy consumption and energy price in Spain. The dataset can be found here. The dataset contains the electrical demand, the generation by type, the prices and the weather in Spain. It includes also the forecast from the Spanish Transmission Service Operator, so that you can compare your results with a real model.

You can choose the regression method that you prefer, but you should be able to justify the choice of the regression model, as well as the choice of the model's parameters.

Task 3: (Optional) Build a Recurrent Neural Network (RNN) for the forecast of the daily energy consumption and energy price. The RNN is a special type of neural network that is particularly suited for time-series forecasting because it is able to capture temporal relationship in the data.

Written report

Your written report should contain the following:

Section 1: A brief introduction of the mathematical formulation of polynomial regression and neural networks. In particular, explain how the different loss functions influence the behaviour of the models. This section should include also a discussion regarding the risk of overfitting, and which strategies can be adopted to mitigate this risk.

Section 2: Results corresponding to **Tasks 1**, compared to the analytical solution. Each result has to be justified by referring to the mathematical formulation of the model.

Section 3: Results corresponding to **Tasks 2**, including relevant figures.

Section 4: Final discussion and conclusions, highlighting the different situations in which it is preferable to use one model over the others. Don't forget to cite the relevant literature to strengthen your conclusions.

Resources

The Scikit-Learn contains most of the linear and non-linear regression models that you can use. In particular, the User Guide offers a good introduction on the mathematical formulation of the linear regression techniques, and it can be used to access the functions' documentation.

You can find a more in depth discussion of linear regression and shrinkage methods in the third chapter of the book "The Elements of Statistical Learning" [1].

In case you are building a neural network as your regression model, you can use either `PyTorch` (more complex but highly flexible) or `TensorFlow` (mare black box type of software). Also, if you are using `Colab`, you can request GPU resources for the training.

If you encounter problems in the download of the data or installation of the libraries you can contact Alberto Procacci on TEAMS or at alberto.procacci@ulb.be.

References

[1] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2009.