Coursework 2: Mathematics for Machine Learning

Romain REINERT

November, 17th 2020

1 Linear Regression

1.1 Question a: polynomial basis

As explained in the section "Basis Functions", we will use the notation $\phi(x) = \begin{bmatrix} \phi_0(x) \\ \phi_1(x) \\ \phi_2(x) \\ ... \\ \phi_M(x) \end{bmatrix} = \begin{bmatrix} 1 \\ x \\ x^2 \\ ... \\ x^M \end{bmatrix}$

Using the theory of the book, we compute the maximum likelihood for $\mathbf{w}$ using the formula :

$$\mathbf{w}_{\text{ML}} = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{y}$$

where, for n points expressed in a polynomial basis of order M:

$$\mathbf{\Phi} = \begin{bmatrix} \phi(x_1) & \phi(x_2) & ... & \phi(x_n) \end{bmatrix}^T = \begin{bmatrix} 1 & x_1 & x_1^2 & ... & x_1^M \\ 1 & x_2 & x_2^2 & ... & x_2^M \\ ... & ... & ... & ... & ... \\ 1 & x_n & x_n^2 & ... & x_n^M \end{bmatrix} \in \mathbb{R}^{N\times(M+1)}$$

This formula is obtained by considering the log-likelihood $\log p(\mathbf{y}|\mathbf{X}, \mathbf{w})$ and computing its gradient w.r.t $\mathbf{w}$. It is maximised by finding $\mathbf{w}_{\text{ML}}$ such that this gradient is $\mathbf{0}$ (necessary condition). For σ^2, the maximum likelihood estimate is computed using the gradient of the log-likelihood w.r.t σ^2 and equaling it to 0. It is given by the formula:

$$\sigma^2_{\text{ML}} = \frac{1}{N}\sum_{n=1}^{N}(y_n - \phi^T(x_n)\mathbf{w})^2$$

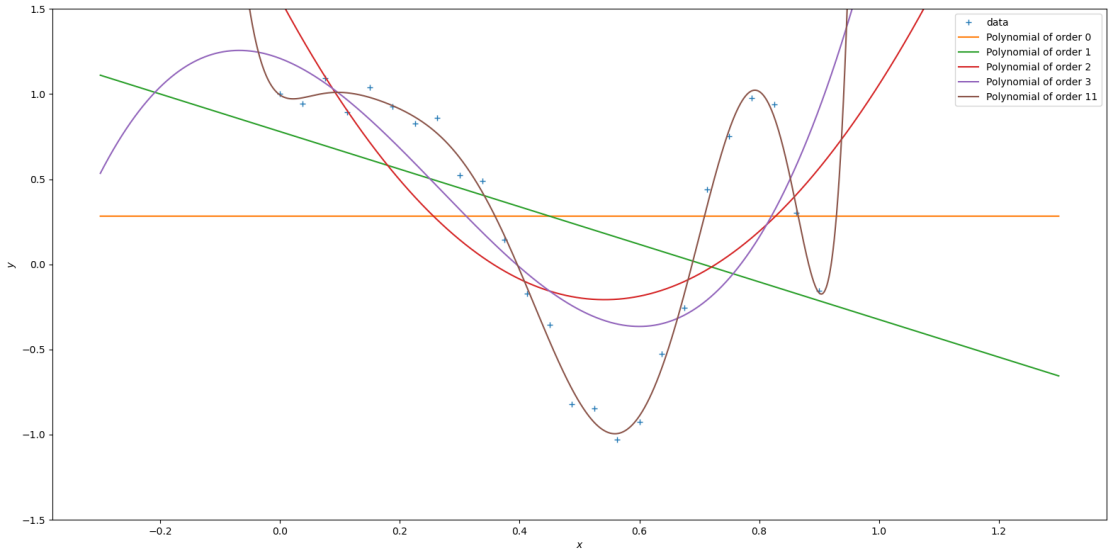

Figure 1: Predicted mean at test points in [-0.3,1.3] for polynomial basis functions of different orders

In the python document, the matrix $\mathbf{\Phi}$ is created using *phi_polynomial*(). The estimator of $\mathbf{w}$ is computed in the *max_lik_estimate*() function. Then the function *plot_polynomial*() plots the different predictions in [-0.3,1.3] for basis of orders 0, 1, 2, 3 and 11.
As expected, higher orders fit better to the given data, which however may result in an over-fitting model (see question d).

1.2 Question b: trigonometric basis

In this question the new matrix $\mathbf{\Phi}$ is:

$$\mathbf{\Phi} = \begin{bmatrix}\phi(x_1) & \phi(x_2) & ... & \phi(x_n)\end{bmatrix}^T = \begin{bmatrix} 1 & \sin 2\pi x_1 & \cos 2\pi x_1 & ... & \sin 2\pi M x_1 & \cos 2\pi M x_1 \\ 1 & \sin 2\pi x_2 & \cos 2\pi x_2 & ... & \sin 2\pi M x_2 & \cos 2\pi M x_2 \\ ... & ... & ... & ... & ... & \\ 1 & \sin 2\pi x_n & \cos 2\pi x_n & ... & \sin 2\pi M x_n & \cos 2\pi M x_n \end{bmatrix} \in \mathbb{R}^{N\times(2M+1)}$$

I used the functions *phi_trigonometric*(), *max_lik_estimate*() and *plot_trigonometric*() to plot the predictions for basis of orders 1 and 11.

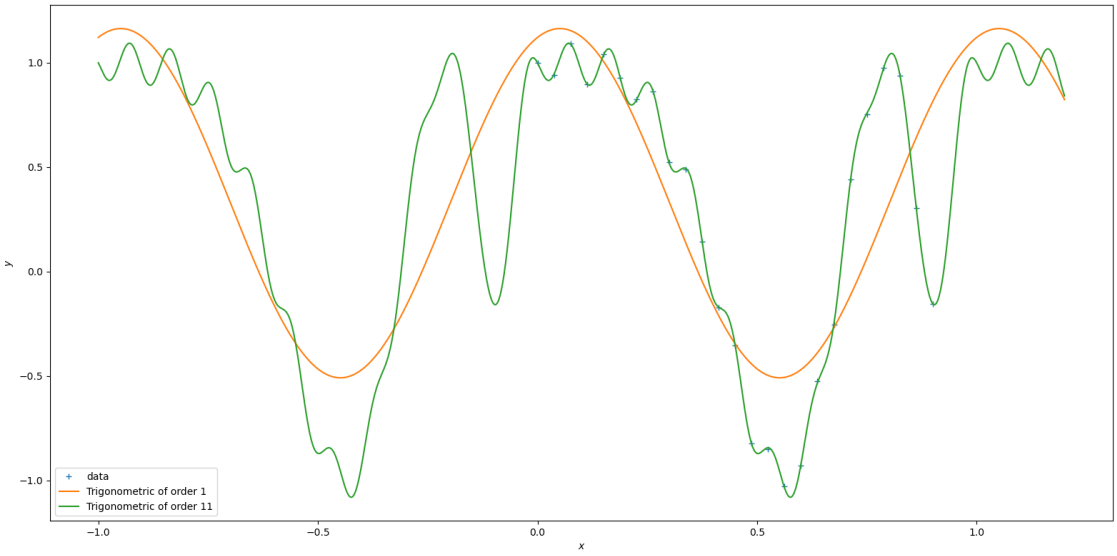

Figure 2: Predicted mean at test points in [-1,1.2] for trigonometric basis functions of orders 1 and 11

1.3 Question c: leave-one-out cross-validation

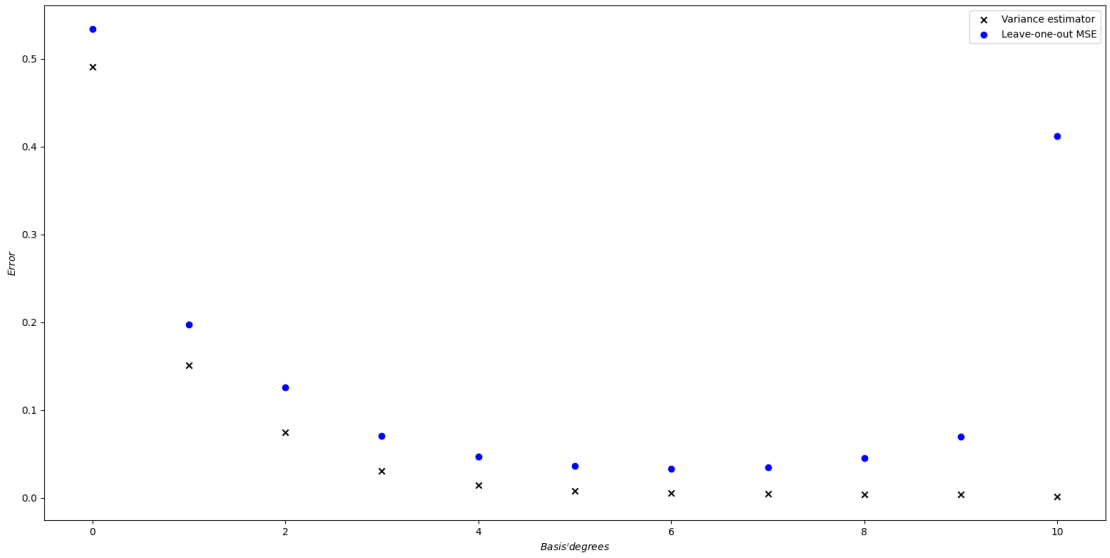

Figure 3: Evolution of the averaged error and the variance estimation using maximum likelihood, against the degree of the trigonometric basis

1.4 Question d: over-fitting

Over-fitting appears when a model fits to a given training data, but doesn't generalize well to unknown data. This can clearly be seen on the previous figure: using leave-one-out cross-validation, we see that while the mean-squared error keeps decreasing on training data as orders of the basis increase from 0 to 10, errors on test data starts to increase for basis of orders superior to 6.
It can be illustrated by the plots of the first two questions, where high degrees polynomials and trigonometric functions show high oscillations to fit exactly to each data-point. The following figure shows that between polynomial basis of orders 4 and 11, weights are increasing widely and successively in opposite signs, in order to allow these strong oscillations fitting perfectly to the data. It suggests that despite a very good fitting to our current points, the model will probably not generalize well to unseen data.

```
w_ml for a polynomial basis of order 4:
[[   0.54204609]
 [  16.83338977]
 [-103.88164731]
 [ 177.58671124]
 [ -92.61235966]]
w_ml for a polynomial basis of order 11:
[[ 9.94382437e-01]
 [-2.42250063e+00]
 [ 8.26455789e+01]
 [-9.73074488e+02]
 [ 5.40927681e+03]
 [-1.55296378e+04]
 [ 1.78595422e+04]
 [ 1.60023399e+04]
 [-7.74854209e+04]
 [ 1.03589628e+05]
 [-6.59935728e+04]
 [ 1.70551217e+04]]
```

Figure 4: Increase in the squared values of the weights

Some solutions can be put in place to try to delay over-fitting. One way to do it is to modify the loss function to penalize the increase in the squared values of weights: this is called regularisation.
An other approach, the Maximum A Posteriori estimation (MAP), uses a prior Gaussian probabilistic distribution on $\mathbf{w}$ to represent the knowledge we have on our weights before having observed any data, thus regularizing their value. Finally, Bayesian Linear Regression (BLR) can be used to give an idea of our confidence in the predictions we are making.