Coursework 2 : Mathematics for Machine Learning (CO-496)

Vincent Jarasse

05/11/2018

1 Linear Regression

1.a

We have

$$P(y \mid x) = \prod_{i=1}^{N} P(y_i \mid x_i)$$

and

$$(\omega, \sigma^2) \in \arg\max_{\omega,\sigma^2} P(y \mid x)$$

Thus

$$(\omega, \sigma^2) \in \arg\min_{\omega,\sigma^2} -log(P(y \mid x)) = -\sum_{i=1}^{N} log(P(y_i \mid x_i))$$

Because

$$y \sim \mathcal{N}(\omega^T \phi(x_i),\, \sigma^2)$$

Then we have

$$\begin{aligned} -log(P(y_i \mid x_i)) &= -log(\frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{(y_i - \omega^T\phi(x_i))^2}{2\sigma^2})) \\ &= \frac{1}{2\sigma^2}(y_i - \omega^T\phi(x_i))^2 + log(\sqrt{2\pi\sigma^2}) \\ &= \frac{1}{2\sigma^2}(y_i - \omega^T\phi(x_i))^2 + constant_{in\ terms\ of\ \omega} \end{aligned}$$

Then

$$-log(P(y \mid x)) = \frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - \omega^T\phi(x_i))^2 + constant_{in\ terms\ of\ \omega}$$

Which, using the proposed notations for vector y, matrix ϕ and vector ω, is equivalent to

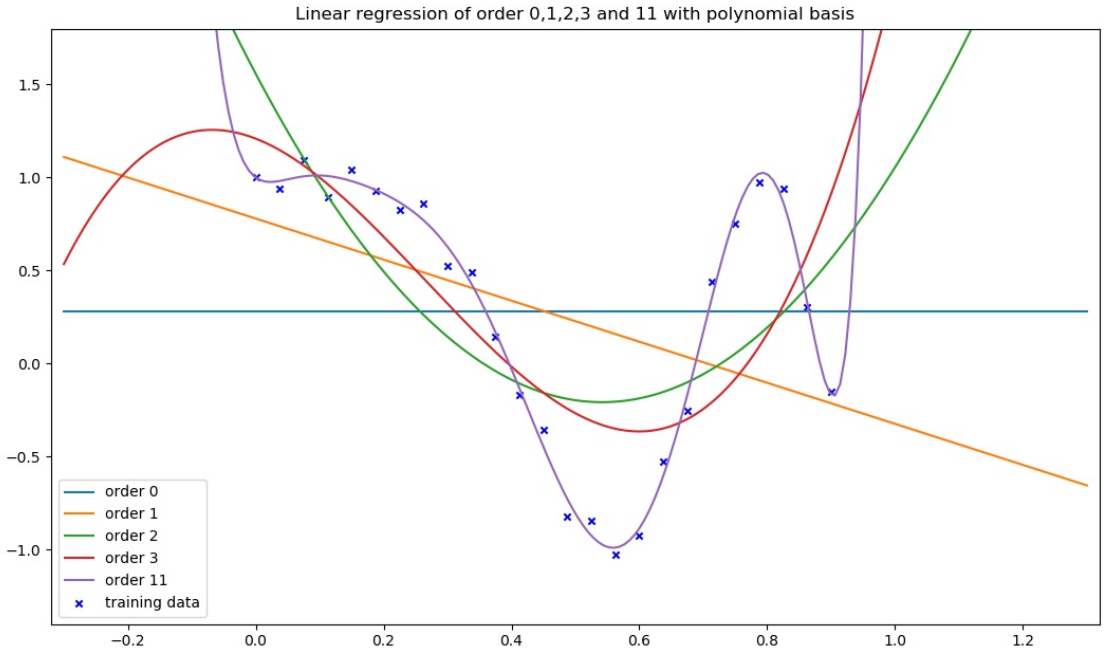
$$-log(P(y \mid x)) = \frac{1}{2\sigma^2}(y - \phi\omega)^T(y - \phi\omega) + constant_{in\ terms\ of\ \omega}$$

Now that we have a matrix expression of the loss function defined as

$$\mathcal{L}(\omega) = -log(P(y \mid x, \omega))$$

we can easily compute its derivative with respect to ω :

$$\frac{\partial \mathcal{L}(\omega)}{\partial \omega} = \frac{1}{\sigma^2}(-y^T\phi + \omega^T\phi^T\phi)$$



We then equal this derivative to 0 to find the ω_{mle} which minimizes the loss function, and thus maximizes $P(y \mid x)$:

$$\frac{\partial \mathcal{L}(\omega_{mle})}{\partial \omega} = 0 \Leftrightarrow \frac{1}{\sigma^2}(-y^T\phi + \omega_{mle}^T\phi^T\phi) = 0 \Leftrightarrow \omega_{mle}^T\phi^T\phi = y^T\phi$$

$$\Leftrightarrow \omega_{mle}^T = y^T\phi(\phi^T\phi)^{-1}$$

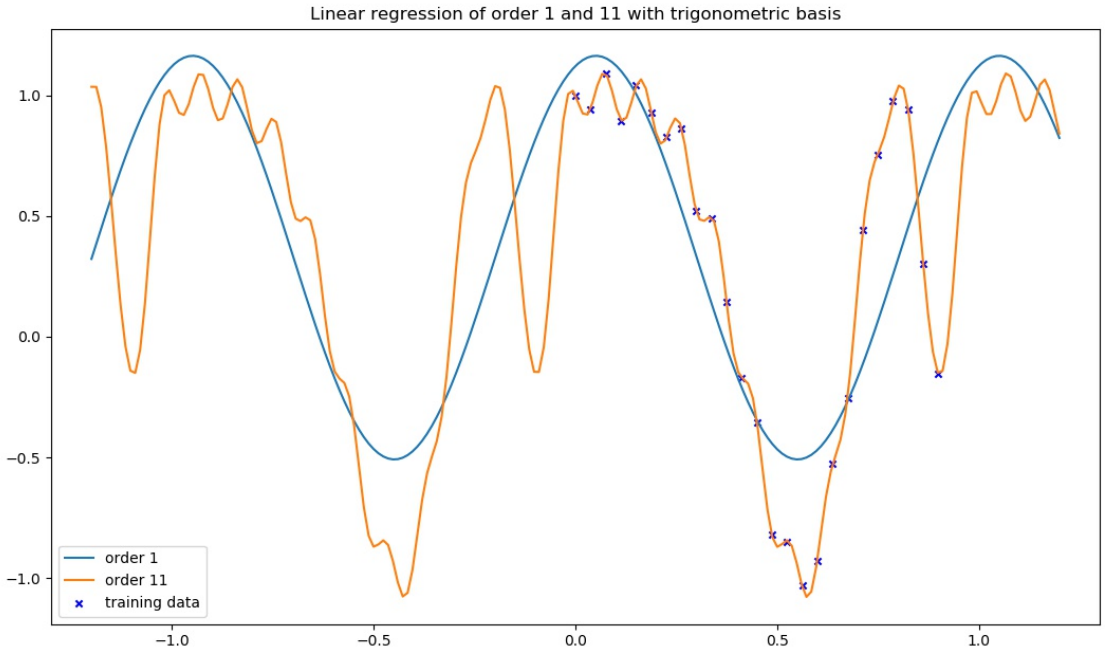
Then we obtain the result ω_{mle} :

$$\boxed{\omega_{mle} = (\phi^T\phi)^{-1}\phi^T y}$$

We can then write a Python program and plot linear regression using this ω_{mle}. Below are the plot of the predictive mean at test points in the interval [-0.3, 1.3] in the case of polynomial basis functions of order 0, 1, 2, 3 and 11. For these plots, we use 200 uniformly spaced points to have smooth curves.

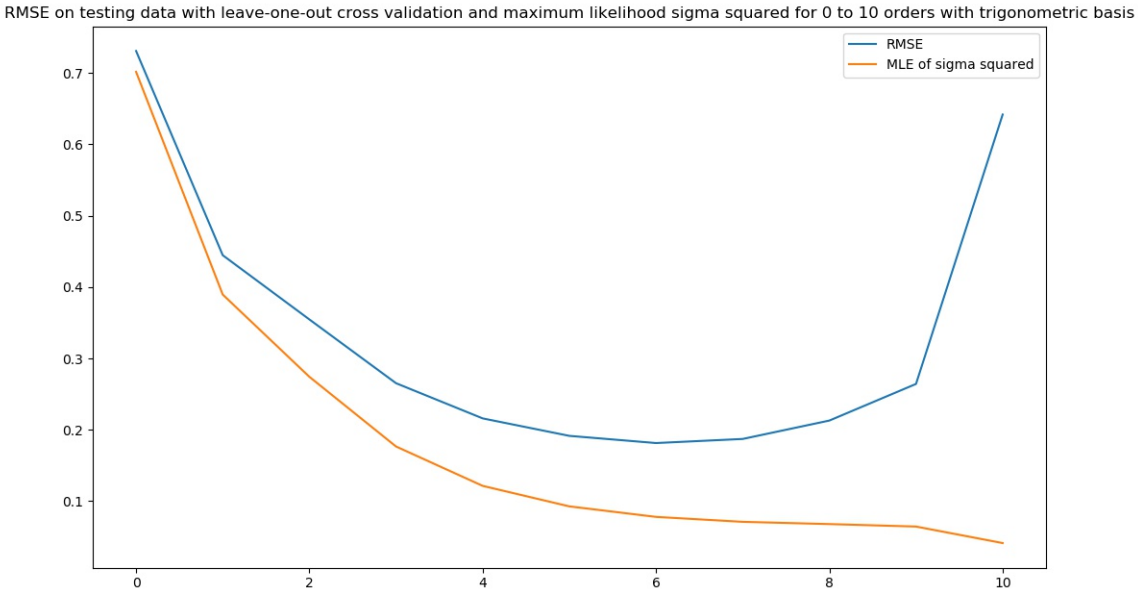
1.b

We repeat the previous part but this time with trigonometric basis functions of orders 1 and 11. Below is the corresponding plot.



1.c

In this question we investigate overfitting, and plot two indicators : RMSE and σ^2 depending on the order of the regression (with trigonometric basis). Below is the corresponding plot.



1.d

In a general way, overfitting is the property of a model which fits the noise structure of the training set : it is a poor representation of the underlying function that has generated the training data.
In our example, we see in plots of question b and c that using higher orders fits the training points better and better, but up to a certain point. Indeed, in question a, the order 11 oscillate wildly, and is very unlikely to represent the underlying function, even if the curve passes through all the training points.
To measure this overfitting property, we plot the graph of question c, on which we have the RMSE using leave-one-out cross validation and the σ^2_{mle} against order of basis. Leave-one-out cross validation allows us to measure the generalisability of our model (in this case the order of the basis used) : we train the model on 24 out of the 25 points, and we compute the error between the predicted value for the 25th and the real value. We do so for each of the 25 points, and average the result, order by order. In a few words, if this indicator is high, it means that our model does not match well the underlying function.
A slightly different indicator is the σ^2_{mle}, which is the maximum likelihood estimation for the variance of the gaussian distribution that y values follow. The more our model approaches training points, the lesser this value is.

If we take a look at the question c plot, we clearly observe two tendances :
- From order 0 to approximately 6, both RMSE and σ^2_{mle} decrease, which mean that our model represent more precisely the underlying function.
- From order 6 to higher orders, RMSE increase quite rapidly whereas σ^2_{mle} still decreases. This means that our model approaches more the training points, but does not generalises.

These observations are consistent with the shapes of the questions a and b plots : with order 11, the graph oscillate wildly, and thus the error between the predicted value and the real value is high on average, even if it is very low for the training points. In our example, we could say that the best

compromise between error and generalisability is order 6 or 7.

2 Ridge Regression

2.a

Let's compute the MAP estimate for ω. Bayes' theorem gives :

$$P(\omega \mid x, y) = \frac{P(y \mid x)P(\omega)}{P(x \mid y)}$$

From which we obtain

$$-log(P(\omega \mid x, y)) = -log(P(y \mid x)) - log(P(\omega)) + constant_{in\ terms\ of\ \omega}$$

We recognise in the first term of the right part the negative log likelihood, which we have already computed in part 1.
The second term is the negative log of the prior. Let's take $\omega \sim \mathcal{N}(0, b^2 I)$. We then have :

$$P(\omega) = (2\pi)^{-D/2} det(b^2 I)^{-1/2} exp(-\frac{1}{2}\omega^T (b^2 I)^{-1}\omega)$$

with ω vector of size D. Then we obtain :

$$-log(P(\omega)) = \frac{1}{2b^2}\omega^T\omega + constant_{in\ terms\ of\ \omega}$$

Therefore, we have :

$$-log(P(\omega \mid x, y)) = \frac{1}{2\sigma^2}(y - \phi\omega)^T(y - \phi\omega) + \frac{1}{2b^2}\omega^T\omega + constant_{in\ terms\ of\ \omega}$$

If we compute the derivative of this expression with respect to ω we obtain :

$$\frac{\partial - log(P(\omega \mid x, y))}{\partial \omega} = \frac{1}{\sigma^2}(-y^T\phi + \omega^T\phi^T\phi) + \frac{1}{b^2}\omega^T \quad (i)$$

Now let's take the loss function provided, express it in terms of vectors and compute its derivative with respect to ω. We obtain

$$\frac{\partial \mathcal{L}(\omega)}{\partial \omega} = 2(-y^T\phi + \omega^T\phi^T\phi) + 2\lambda\omega^T \quad (ii)$$

Both expressions (i) and (ii) can be equaled to 0 and we have the correspondance between the two by setting :

$$\boxed{\lambda = \frac{\sigma^2}{b^2}}$$

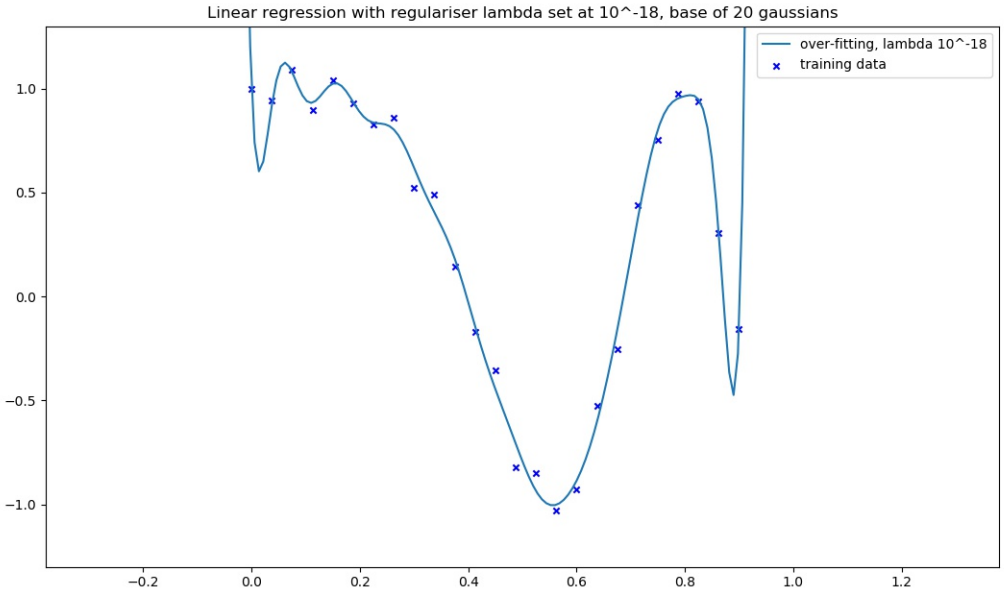
and

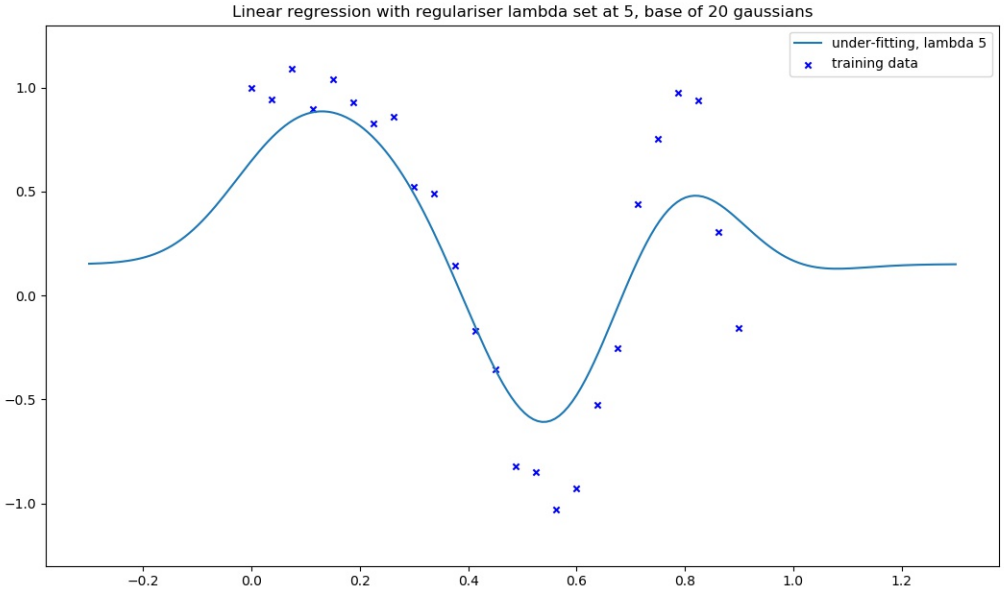
$$\boxed{\omega_{map} = (\phi^T\phi + \lambda I)^{-1}\phi^T y}$$

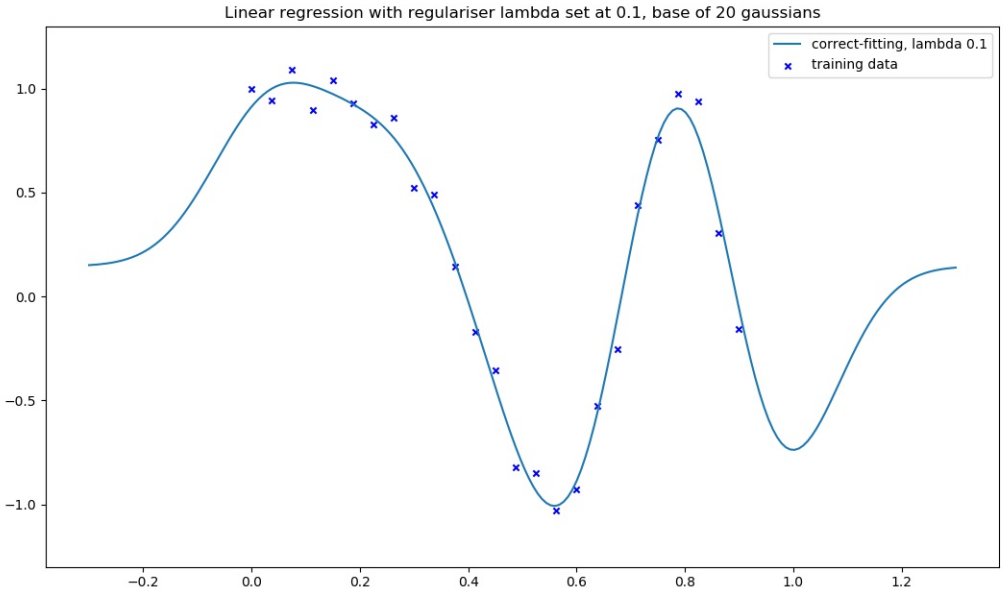
This expression of the loss function is pretty clear : because we want to minimise the loss function, we need to minimise its second term. This second term will be big when high parameters are high. By stating that the prior follows a normal distribution centred around 0 and of variance $b^2 I$, we lower the impact of high order parameters. We now understand that λ is the "strictness" of the regularisation.

2.b

Below are the plots using the regularised least square value for ω, with 3 different values for λ. The first plot is $\lambda = 10^{-18}$, which is almost 0 and then shows over-fitting like in part 1. The second plot uses $\lambda = 5$ and clearly shows under-fitting : the regulariser is too high and we lose the underlying function. The third plot is $\lambda = 0.1$ and is quite satisfying.


Linear regression with regulariser lambda set at 10^-18, base of 20 gaussians
over-fitting, lambda 10^-18
training data
1.0
0.5
0.0
−0.5
−1.0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2


Linear regression with regulariser lambda set at 5, base of 20 gaussians
under-fitting, lambda 5
training data
1.0
0.5
0.0
−0.5
−1.0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2


Linear regression with regulariser lambda set at 0.1, base of 20 gaussians
correct-fitting, lambda 0.1
training data
1.0
0.5
0.0
−0.5
−1.0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2