Mathematics for Machine Learning - Coursework 2

Constantin Eulenstein

November 2020

1 Linear Regression

1.1 Exercise 1a)

In the specifications we are told that y_i is distributed according to a gaussian linear model with mean $\boldsymbol{w}^T\boldsymbol{\phi}(x_i)$ and variance σ^2 ($y_i \sim \mathcal{N}\left(\boldsymbol{w}^T\boldsymbol{\phi}\left(x_i\right), \sigma^2\right)$); consequently, the likelihood $p(y_i|x_i)$ is normally distributed. Our aim is to maximise the probability of generating the particular data Y that we observed w.r.t the parameters $\boldsymbol{w}$ of our model. In the following I will derive the maximum likelihood solution of $\boldsymbol{w}$ and σ^2 in terms of the design matrix Φ, which is constructed as explained in the specifications.

$$\boldsymbol{w}^*, \sigma^{*2} = \underset{\boldsymbol{w},\sigma^2}{\operatorname{argmax}}\, p(\mathbf{y} \mid \boldsymbol{w}, X) = \underset{\boldsymbol{w},\sigma^2}{\operatorname{argmax}} \prod_{i=1}^{N} p\left(y_i \mid \boldsymbol{w}, \mathbf{x}_i\right) = \underset{\boldsymbol{w},\sigma^2}{\operatorname{argmax}} \sum_{i=1}^{N} \log p\left(y_i \mid \boldsymbol{w}, \mathbf{x}_i\right) =$$
$$= \underset{\boldsymbol{w},\sigma^2}{\operatorname{argmax}} \sum_{i=1}^{N} \log p\left(y_i \mid \boldsymbol{w}, \mathbf{x}_i\right) = \underset{\boldsymbol{w},\sigma^2}{\operatorname{argmax}} \sum_{i=1}^{N} \log\left[(\sqrt{2\pi}\sigma)^{-1} \exp\left[-\left(y_i - \boldsymbol{w}^T\boldsymbol{\phi}\right)^2 / 2\sigma^2\right]\right] =$$
$$= \underset{\boldsymbol{w},\sigma^2}{\operatorname{argmax}} -\tfrac{N}{2}\log(2\pi\sigma^2) - \textstyle\sum_{i=1}^{N}\left(y_i - \boldsymbol{w}^T\boldsymbol{\phi}\right)^2 / 2\sigma^2 = \underset{\boldsymbol{w},\sigma^2}{\operatorname{argmax}} -\tfrac{N}{2}\log(2\pi\sigma^2) - \|\boldsymbol{Y} - \boldsymbol{\Phi}\boldsymbol{w}\|^2 / 2\sigma^2$$

We first start by solving the log likelihood for the optimal $\boldsymbol{w}^*$, since its solution will be independent of σ^2. Consequently, we take the derivative w.r.t $\boldsymbol{w}$ and set it equal to 0:

$$\tfrac{\partial}{\partial \boldsymbol{w}} -\tfrac{N}{2}\log(2\pi\sigma^2) - \|\mathbf{Y} - \boldsymbol{\Phi}\boldsymbol{w}\|^2 / 2\sigma^2 = -\tfrac{1}{2\sigma^2}\tfrac{\partial}{\partial \boldsymbol{w}} \|\mathbf{Y} - \boldsymbol{\Phi}\boldsymbol{w}\|^2 = -\tfrac{1}{2\sigma^2}(\mathbf{Y} - \boldsymbol{\Phi}\boldsymbol{w})^\top(-\boldsymbol{\Phi}) =$$
$$= \tfrac{1}{2\sigma^2}\boldsymbol{\Phi}^\top(\mathbf{Y} - \boldsymbol{\Phi}\boldsymbol{w}) = 0$$

From the last equation follows that $\boldsymbol{\Phi}^\top\mathbf{Y} = \boldsymbol{\Phi}^\top\boldsymbol{\Phi}\boldsymbol{w}$, which results in the optimal parameter estimate $\boldsymbol{w}^* = (\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top\mathbf{Y}$. To test whether this is actually a maximum we can take the second derivative of our log likelihood w.r.t. $\boldsymbol{w}$ and show that our Hessian matrix $\mathbf{H}$ is negative semidefinite:

$$\mathbf{H} = \tfrac{\partial}{\partial \boldsymbol{w}} \tfrac{1}{2\sigma^2}\boldsymbol{\Phi}^\top(\mathbf{Y} - \boldsymbol{\Phi}\boldsymbol{w}) = -\tfrac{1}{2\sigma^2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$$

Since $\sigma^2 > 0$ and $\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$ is a symmetric matrix, it follows that $\boldsymbol{v}^\top\mathbf{H}\boldsymbol{v} \leq 0, \forall \boldsymbol{v}$; so indeed we found a maximum. Lastly, our individual predicted means become $\boldsymbol{w}^{*T}\boldsymbol{\phi}(x_i)$ (and the predicted means vector becomes $\boldsymbol{\Phi}\boldsymbol{w}^*$).

To then find the maximum likelihood solution for σ^2, we find $\underset{\sigma^2}{\operatorname{argmax}} -\tfrac{N}{2}\log(2\pi\sigma^2) - \|\boldsymbol{Y} - \boldsymbol{\Phi}\boldsymbol{w}\|^2 / 2\sigma^2$ by taking the derivative w.r.t σ^2 and setting it equal to 0: $-\tfrac{N}{2}\sigma^{-2} + \tfrac{1}{2}\sigma^{-4}\|\boldsymbol{Y} - \boldsymbol{\Phi}\boldsymbol{w}\|^2 = 0$. By solving this equation and plugging in our recently found optimal parameter estimate $\boldsymbol{w}^*$ one gets $\sigma^{*2} = \tfrac{1}{N}\|\boldsymbol{Y} - \boldsymbol{\Phi}\boldsymbol{w}^*\|^2 = \tfrac{1}{N}\left\|\boldsymbol{Y} - \boldsymbol{\Phi}(\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top\mathbf{Y}\right\|^2$.

Summarized, by calculating my design matrix Φ depending on the order K of the basis functions one uses, one can find $\boldsymbol{w}^*$. In the case of polynomial basis functions, this yields the predicted means for different orders K: $\sum_{j=0}^{K} w_j^* x^j$. In figure 1 I show the curves derived from polynomial basis functions of different orders including the data (red crosses). One can see that with rising order, the curve of the predicted means predicts the training data better and better. The maximum likelihood estimator (MLE) in the case of polynomial basis functions of order 11, predicts the data especially well on the interval [0,0.9], but shoots off towards very large positive values directly before and after the training interval (I actually have to restrict the y-axis such that the plot does not only show the offshoot). Consequently, if the estimator should predict values that are also outside the training data interval, it will do very poorly.

To overcome this problem of shooting off outside the interval [0,1] (what especially polynomials of high order often do) we can use trigonometric basis functions.

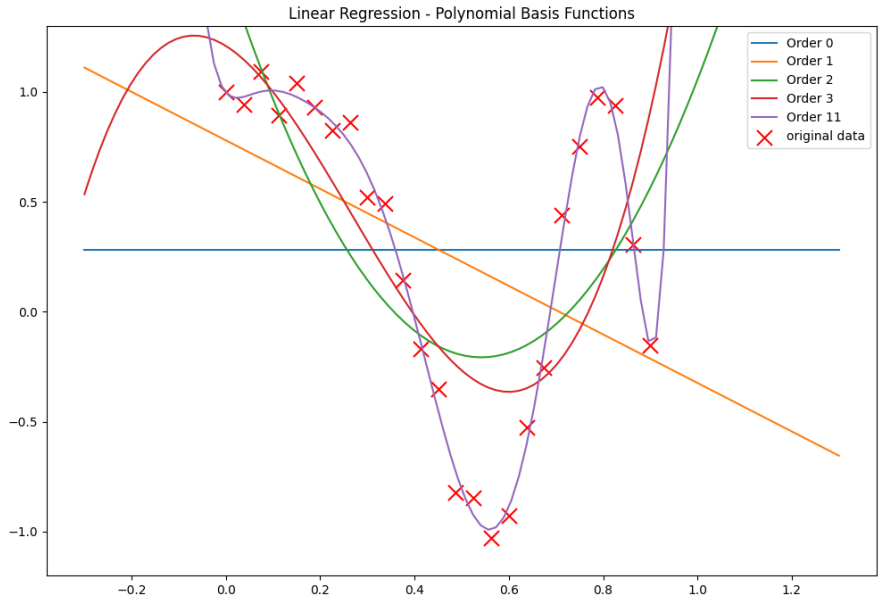

Figure 1: Curves of the predicted means in the case of polynomial basis functions of different orders and training data

1.2 Exercise 1b)

Same as in Exercise 1a), I am plotting the curves of the predicted means in figure 2 for different orders K: $\sum_{j=0}^{K} w_{2j-1}^{*} sin(2\pi jx) + w_{2j}^{*} cos(2\pi jx)$. I also plot the training data in form of red crosses. However, this time the design matrix consists of the data, evaluated using trigonometric basis functions. Consequently, we are also approximating the data using these trigonometric basis functions. One can clearly see how a MLE in the case of trigonometric basis functions of order 11 does not shoot off and fits the training data even better than a MLE in the case of polynomial basis functions of order 11. This makes sense, since we know that the training data has been created using a trigonometric function. Also the plots show how the MLE in the case of trigonometric basis functions of order 1 does not fit the data well, but still way better than a MLE in the case of polynomial basis functions of order 1 (which was just a straight line $w_0 + w_1 x$).

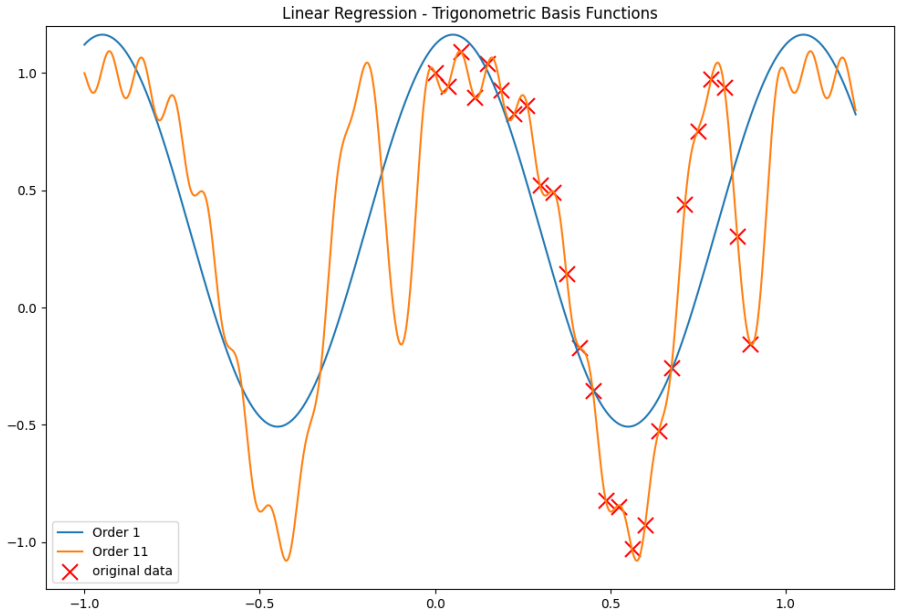

Figure 2: Curves of the predicted means in the case of trigonometric basis functions of different orders and training data

1.3 Exercise 1c)

In figure 3, I depict two graphs that are both plotted against the order K of trigonometric basis functions. The orange graph shows the average squared error that has been obtained by doing leave-one-out cross validation of the training data and averaging over the squared error that was obtained with each fold (25 folds). To run leave-one-out cross validation, we build the design matrix (according to order K) with only 24 training points (all except for the point that is used for validation). From the design matrix and our labels of the 24 training points we can calculate our parameters $\boldsymbol{w}^*$ and, therefore, obtain the function of the predicted means in the case of trigonometric basis functions. Now we can predict the output of our 25th test point, when evaluated with this trigonometric function, and estimate the squared error w.r.t. the left out data label $(y_{predicted} - y_{validation})^2$. This is done until every data point has been left out once and the average over the 25 errors is calculated, which is shown for cases of trigonometric basis functions of different orders K. Additionally, the average σ^2 is plotted for different orders K in the blue graph. In each fold of the cross validation a σ^2 is calculated via $\frac{1}{N} \|\boldsymbol{Y} - \boldsymbol{\Phi}\boldsymbol{w}^*\|^2$, where N is equal to 24, $\boldsymbol{Y}$ corresponds to the 24 training data labels and $\boldsymbol{\Phi}$ is the design matrix built from the 24 data points (here again $\boldsymbol{\Phi}$ and $\boldsymbol{w}^*$ are dependent on K). The average over the 25 found σ^2 is plotted against each order.

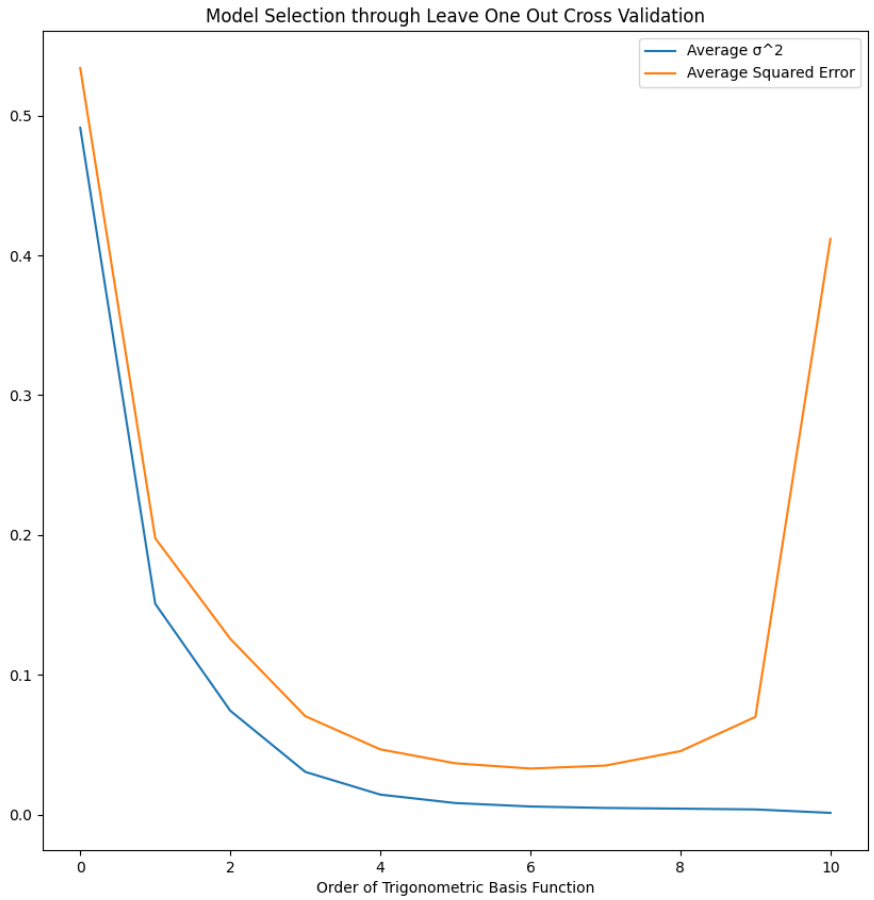

Figure 3: Averaged square error of leave-one-out cross validation and averaged σ^2 over the order of trigonometric basis function

1.4 Exercise 1d)

Overfitting means that our estimator is performing very well on our training data (e.g. fits the training data perfectly) but very poorly on unseen test data.

For a qualitative discussion, we can analyze figures 1 and 2. In both figure 1 and 2 we can see that MLEs of low orders (independent whether trigonometric or polynomial) fit the training data rather poorly (underfitting). With rising order, the MLEs start fitting the training data better. Especially, the order 11 MLEs fit the training data for both cases very well. But, as described before, the MLE in the case of polynomial basis functions of order 11 shoots off on any interval that it has not been trained on. This shows us that it is heavily overfitting the training data and will generalize very poorly on new data. This does not occur in such an extent for MLEs with trigonometric basis functions since they are periodic. However, one can clearly see that, for order 11, the trigonometric function tries to exactly fit the data in the interval [0,0.3], which is where the zigzagging occurs. This is also a potential sign of overfitting. In general over the entire interval [0,0.9] the function seems to fit every training data point almost perfectly, but interpolates between them rather poorly.

A more fundamental and quantitative discussion of whether our trigonometric MLE overfits and which orders are especially sensitive to overfitting, gives figure 3 (it thereby is a good evaluation for model selection). First of all, I want to clarify what the average squared error and average σ^2 signify from my understanding. Since we are only given very few training data and no test data, we use leave-one-out cross validation for evaluating whether our MLE generalizes well on unseen data. The average squared error gives us an approximate of how well our MLE will work on new unseen data that has been obtained from the same distribution like the training data. The average variance σ^2 of our MLE, on the other hand, will tell us overall how well we are fitting our training data (and whether there is a high spread

between the errors for individual training data points). Consequently, while the average squared error gives us an estimate of our MLE on unseen data, the variance σ^2 gives us only an estimate on already seen training data.

Now, the average squared error improves (decreases) continuously, when using higher order basis functions up to order 6 inclusive. For our training data, a MLE obtained from trigonometric basis functions of order 6 would generalize and fit previously unseen data best. With orders higher than 6, our average square error increases, which shows us that overfitting starts at order 7, and gets worse with every higher order. Because of the higher order trigonometric function, our estimator has more possibilities to fit our training data better, which usually comes with worse interpolation between data points.

With rising orders, our average variance σ^2 keeps on decreasing continuously and becomes overconfident with high orders. Because we are fitting our training data better and better, our MLE's estimate of σ^2 converges towards 0 until actually every training data point is passed by our MLE directly and σ^2 becomes 0.

Consequently, when our MLE actually begins to overfit with rising order (order 7 and up), it also becomes more and more confident on the training data (which implies the continuously decreasing σ^2). But, the average squared error from leave-one-out cross validation increases and, therefore, tells us that our MLE generalizes actually worse and not better. From figure 3 one can see that leave-one-out cross validation is a suitable technique for model selection. From the figure, we can choose the model that minimizes our average square error (order 6). Otherwise, to prevent overfitting on higher orders we'd have two options: 1) Simply collect more training data to obtain a smoother curve. 2) Introduce regularization term for our loss function to penalize large parameter values and discourage bad solutions.