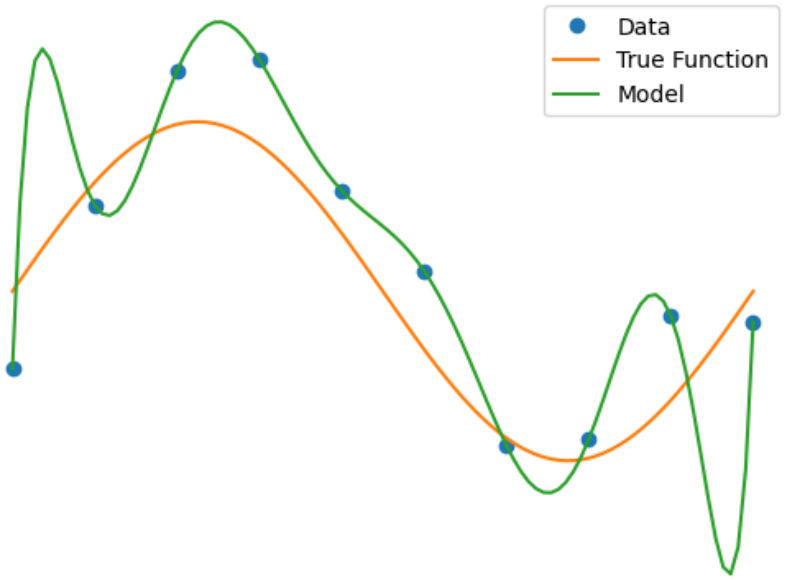
Regularizing Regression Models

Axel Orrhede

October 2024



1 The problem

In this paper, we will try to fit a regression model to a sinusoidal function where the data has normally distributed noise according to equation 1.

$$y_i = \sin(2\pi x_i) + \epsilon\mathcal{N}(0, 1), \text{where } \epsilon > 0 \tag{1}$$

If we limit ourselves to polynomial curve-fitting with a basis of monomials by minimizing squared errors, we can produce 10 data points and create figure 1.

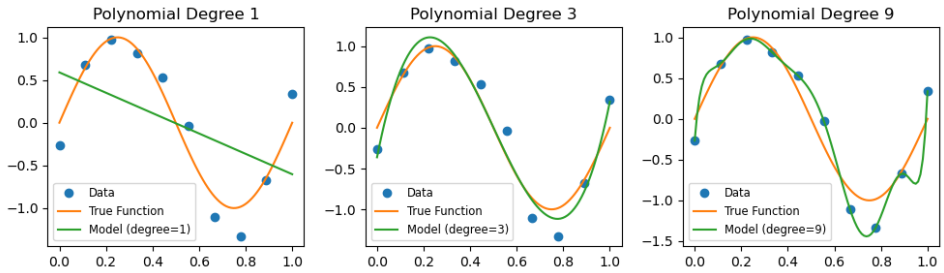


Figure 1: Polynomial curve fitting with basis of monomials up to degree 1,3 and 9

From this, we find that even though higher order polynomials should allow us to recreate a sinusoidal function better, e.g. the Taylor Series of sinus, we do not get better results by having higher order polynomials. In fact, if we look at the Mean Squared Error for the regression seen in figure 1, we can see that the third degree seems to be the best we can do, see figure 2.

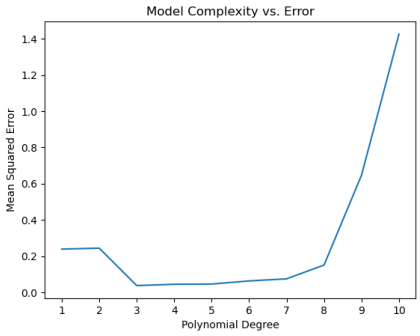


Figure 2: The mean squared error to the true function over the degree of polynomial used

These results may at first glance seem surprising since third-degree polynomials are a subset of 9th-degree polynomials. For this reason, it should at least perform as well as the third-degree one. The problem here is overfitting. We have 10 data points in

the training data, and therefore a polynomial of degree 9 can fit to them perfectly, which is the behavior we see in the rightmost plot of figure 1.

2 Regularization

Previously, we have minimized the ordinary least squares equation 2.

$$L(w) = ||Xw - y||^2 \tag{2}$$

Where X is the design matrix with n rows, one for each data point, and $m + 1$ columns for polynomials of degree m. If we add a regularization term to this that penalizes large coefficients (w), we limit the overfitting behavior seen in the previous section.

$$L(w) = ||Xw - y||^2 + \lambda||w|| \tag{3}$$

This gives us the following normal equation that we can solve to hopefully get a better regression.

$$w = (X^T X + \lambda I)^{-1} X^T y \tag{4}$$

What remains is to find the value of λ.

3 Calculating the optimal value of λ

In *Pattern Recognition and Machine Learning* Christopher Bishop notes that $\lambda = e^{-18}$ is a good value to regularize a polynomial fitting much like the rightmost graph in figure 1[1]. Let us see if this value is the best one.

To do this, we need a way to quantify a "good fit". This will be measured by sampling 100 evenly spaced in the domain and calculating the mean squared distance between the model and the true function. This allows us to set up a minimization with a standard optimization library such as Scipy with hopes of finding the optimal λ. Sadly, this will yield severely inconclusive results.

It turns out that the optimal λ is too heavily dependent on the noise in the data generation. In the leftmost graph in figure 3 we can easily find an optimal lambda to minimize loss. In the other graph, however, we see another behavior. This time, new data points have been sampled every time, and now the correlation between λ and loss is too weak to find an optimum.

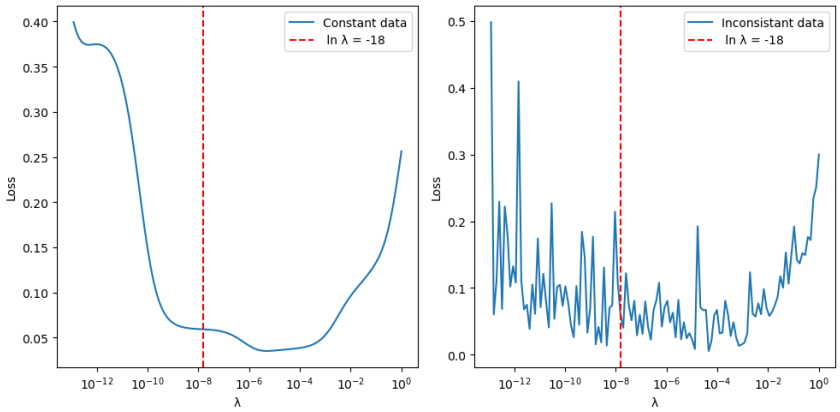


Figure 3: Loss over λ for polynomials of degree 9 fitting to 10 data points. The right graph has constant data points, the left one has newly sampled data for each regression.

This makes finding a "one lambda fits all" an impossible task, even if we limit ourselves to predicting a single period of the sinus function with 10 data points with polynomials of degree 9.

Before we give up, we shall try to indicate a good starting spot for picking λ. Let us see if we find any trends by performing multiple regressions and averaging them out. While we are at it, we will also try to modify some parameters to see if our findings can be generalized.

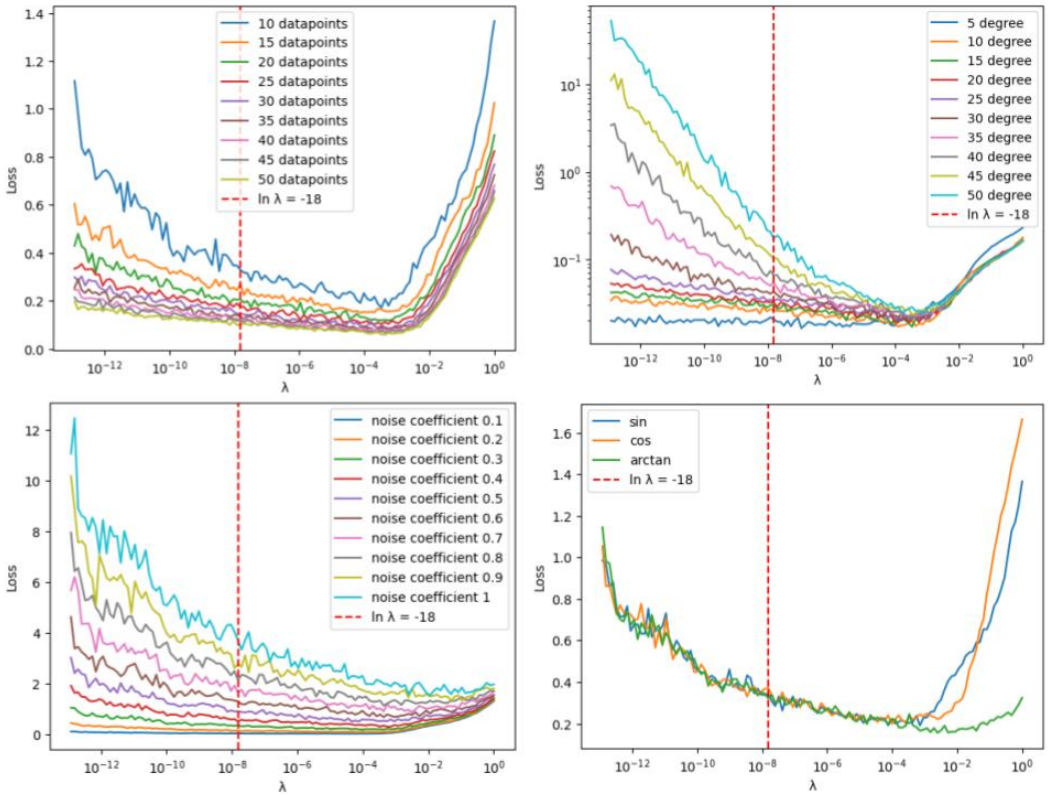


Figure 4: Plots similar to the right graph in Figure 3, but the graphs are averaged over 100 different iterations with different data sets. The upper left image has a different number of data points and polynomials of 1 degree less. The upper right image has a different number of degrees of polynomial fitted to 25 data points, the lower left image has different noise coefficients and the lower right image has different functions fitted to it.

In these graphs, we can see that there is a sweet spot for λ, just as Bishop describes it [2]. However, the sweet spot is not even close to e^{-18}. Instead figure 4 shows us that $\lambda = 10^{-3}$ seems to be close to the minimum in most of these cases. As mentioned before, we need to look at this number with a lot of caution. As seen in figure 3, every set of data points will have its optimal value for λ

4 A λ-selection procedure

Another method for choosing λ using Leave-One-Out Cross-Validation is described by Golub in *Matrix Computations* [2]. We start by removing one of the data points

from our modified normal equation 3 by using D_k which is an identity matrix but with a 0 on the k^{th} diagonal element.

$$L_{cv}(w) = ||D_k(Xw_k - y)||_2^2 + \lambda||w||_2^2 \tag{5}$$

Minimizing this loss allows us to calculate the coefficients with the k^{th} data point left out, which we will denote w_k. Repeating this process for all data points allows us to calculate a cross-validation weighted (v) square error $C(A)$.

$$C(\lambda) = \frac{1}{n}\sum_{k=1}^{n} v_k(x_k^T w_k(\lambda) - y_k)^2 \tag{6}$$

Where the weights v_k could let us lessen or increase the impact of certain data points, such as on the edges of our domain. The λ we are looking for is the one that minimizes equation 6. Lucky for us, Golub finds a way to easily calculate w_k.

$$w_k(\lambda) = w(\lambda) + \frac{x_k^T w(\lambda) - y_k}{1 - z_k^T x_k} z_k \tag{7}$$

Where $z_k = (X^T X + \lambda I)^{-1} x_k$ and $w(\lambda) = (X^T X + \lambda I)^{-1} X^T y$. Using this equation and a SVD transform $X = U\Sigma V^T$, Golub manipulates equation 6 into the following.

$$C(\lambda) = \frac{1}{n}\sum_{k=1}^{n} v_k \left[\frac{u_k y - \sum_{j=1}^{n} u_{kj}^2 y_j \left(\frac{\sigma_j^2}{\sigma_j^2 + \lambda} \right)}{1 - \sum_{j=1}^{n} u_{kj}^2 \left(\frac{\sigma_j^2}{\sigma_j^2 + \lambda} \right)} \right]^2 \tag{8}$$

Minimizing this equation should allow us to find the optimal λ for our data.

Calculating $C(\lambda)$ has a time complexity of $O(n^2)$ but minimizing will differ based on your chosen method. Instead, we can compare this to solving the problem and calculating the normal loss function. There, the time-consuming steps in calculating $L(\lambda$ is the $X^T X$ multiplication with time complexity of $O(n^2 m)$ and inverting $X^T X + \lambda I$ with time complexity $O(m^3)$. This means that minimizing $C(\lambda)$ will always be faster.

References

[1] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[2] G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 4. JHU Press, 2013.