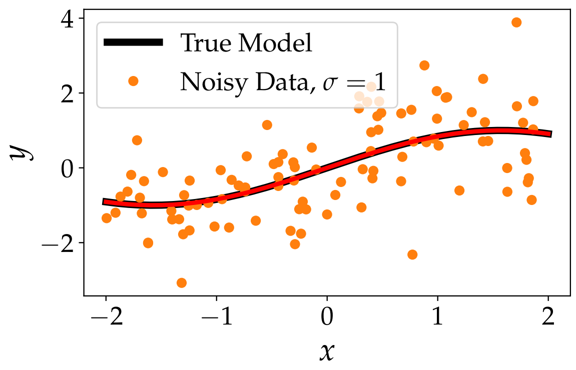
Statistics, Data Analysis, and Machine Learning for Physicists

Timothy Brandt

Spring 2020

Lecture 16

In this class we'll transition to a new problem: predicting the value of a quantity from a set of noisy measurements in the absence of a proper model. This is hard, and something you would like to avoid if possible. Suppose that you have the following set of measurements:



If you know that these data points are drawn from a sine, then you can fit a model using the machinery we developed earlier in this course. Now we'll assume that you do not know the model from which the data were drawn, but you want to be able to predict its value at a given position. In other words, given the orange data points, what is the value of the model at, say, $x = 1$, and what is my uncertainty on its value?

In order to begin this process, I have to start somewhere. I will assume a *covariance function*. This imposes a relationship between the model at different data points. For example, I might believe that the true model is a smooth function of x, and varies on a characteristic scale (say, $\Delta x = 1$). Alternatively, I might believe that it is smooth, but I might not know what scale it varies over. In that case, I could try different covariance functions and see which one works best (which is another problem that we'll discuss in a little bit). There is no need for the model to be a function of just a single parameter, as long as I can write down a positive definite covariance matrix of scalar covariances for any set of points.

Gaussian process regression takes as its underlying assumption that everything is Gaussian in the following sense. I have a covariance matrix for my true values of the function at all values of x. For example, if the true value of the model at x_1 is $f(x_1) = y_1$, then the function value at x_2, $y_2 = f(x_2)$ should be drawn from a Gaussian with known mean and variance. (Conceptually, for now, just assume that I know the function's value *somewhere*). My covariance function specifies the mean and variance of the function value at x_2 given a value at x_1, with

$$K(x_1, x_2) = \mathrm{Cov}\left(f(x_1), f(x_2)\right) \tag{1}$$

To take a concrete example, suppose that

$$K(x_1, x_2) = 1 \;\forall\; \{x_1, x_2\}. \tag{2}$$

In this case, the function values at x_1 and x_2 are perfectly correlated. If $K(x_1, x_2) = 1$ for all x_1, x_2, then my function must be a constant everywhere. We'll return to this example later to try to get a bit of intuition, even if we'll never use this covariance function in practice.

Now suppose I have measurements of the function at some set of points $\mathbf{x}$, and that I would like to know what the values are at some other points $\mathbf{y}$, and I would like to have estimates for the uncertainties on those predictions. This is what Gaussian process regression gives you. To rephrase, since this is maybe the most important thing to remember about Gaussian process regression,

Gaussian process regression takes in measurements at given parameter values, along with assumptions about the smoothness of the underlying model, and returns predicted model values and uncertainties for arbitrary parameter values.

Gaussian process regression is generally less good than fitting a model, since you are relying on some assumptions about the function (through the adopted covariance) and are not trying to gain physical insight into the model. It is an appropriate choice when you cannot generate a model for your data from underlying physical assumptions.

We'll return to my covariance function, which statistically relates the model value at two parameter values x_1 and x_2. I'll assume that the covariance function is 1 when $x_1 = x_2$ (this enforces that the model must have a single, well-defined value at a given position, since it is perfectly correlated with itself). I will assume that the covariance function is less than 1 when $x_1 \neq x_2$, where (typically) they become less correlated the further apart x_1 and x_2 are. I will then assume, for now, that I *know* the model values at a certain set of n points, call them $\mathbf{X}$, with the model values being $f(\mathbf{X})$. I will further take the covariance between each pair of points x_i in $\mathbf{X}$ to be an $n \times n$ matrix $K(\mathbf{X}, \mathbf{X})$. I then want the model values at the m points $\mathbf{Y}$, $f(\mathbf{Y})$, and the uncertainties on these values.

The basic problem can be written as follows:

$$\begin{bmatrix} f(\mathbf{X}) \\ f(\mathbf{Y}) \end{bmatrix} \sim \mathcal{N}\left[\mathbf{0}, \begin{bmatrix} K(\mathbf{X},\mathbf{X}) & K(\mathbf{X},\mathbf{Y}) \\ K(\mathbf{Y},\mathbf{X}) & K(\mathbf{Y},\mathbf{Y}) \end{bmatrix}\right] \tag{3}$$

In other words, all of my model values are drawn from a joint Gaussian distribution with zero mean and with a covariance matrix defined by K. The initial $\mathbf{0}$ might be a little strange/scary. It is true that having a prior of zero mean is a very real constraint on the function. It is often a good idea to try to make sure that your model, or measurements, have a rough estimate of the mean subtracted before you start. This isn't required in the same sense that it is for principal component analysis, but be aware that Gaussian process regression will tug your fit toward zero.

The first part of Equation (3),

$$f(\mathbf{X}) \sim \mathcal{N}\left[\mathbf{0}, K(\mathbf{X},\mathbf{X})\right], \tag{4}$$

says that my (known) model values at $\mathbf{X}$ were drawn from a vast space of possibilities whose relative likelihoods are defined by how well those functions match the covariance I assumed. As a concrete example, let's return to the covariance function that is just one everywhere. In this case, the only way for all of the $f(\mathbf{X})$ to be drawn from this distribution is if they are all identical. This isn't very useful for points with measurement error. In that case assuming I have a measurement error σ on each point, I would have something like

$$f_{\text{meas}}(\mathbf{X}) \sim \mathcal{N}\left[\mathbf{0}, K(\mathbf{X},\mathbf{X})\right] + \mathcal{N}\left[\mathbf{0}, \sigma^2 \mathbf{I}\right]. \tag{5}$$

My measured points are the real model points plus some error term, and since we are assuming everything to be Gaussian here, the means and variances add. My covariance matrix gets the measurement errors added to it along the diagonal:

$$K(\mathbf{X},\mathbf{X}) \rightarrow K(\mathbf{X},\mathbf{X}) + \sigma^2 \mathbf{I}. \tag{6}$$

More generally, I would have something like

$$K(\mathbf{X},\mathbf{X}) \rightarrow K(\mathbf{X},\mathbf{X}) + \text{Cov}_{\text{data}}(X, X). \tag{7}$$

The first term is the covariance matrix referring to the *model* values, while the second is the good old covariance matrix describing my data.

I'll now state the fundamental property of Gaussians that underpins Gaussian process regression. I haven't found a proof that is sufficiently simple and intuitive that it adds any insight (for me, at least), so I will state this without proof. If the model values and covariance matrices are described by Equation (3), then **the conditional probability distribution of $f(\mathbf{Y})$ given the values $f(\mathbf{X})$ is**

$$f(\mathbf{Y}) \sim \mathcal{N}\left[K(\mathbf{Y},\mathbf{X})K^{-1}(\mathbf{X},\mathbf{X})f(\mathbf{X}),\ K(\mathbf{Y},\mathbf{Y}) - K(\mathbf{Y},\mathbf{X})K^{-1}(\mathbf{X},\mathbf{X})K(\mathbf{X},\mathbf{Y})\right] \tag{8}$$

The distribution of model values at the points $\mathbf{Y}$ is a multivariate Gaussian whose mean and covariance matrix I can compute. This gives me a prescription to compute the mean and variance on each point. If I am including measurement errors, I would add these to $K(\mathbf{X},\mathbf{X})$. The mean and variance of my multivariate Gaussian for $\mathbf{Y}$ would become

$$\boldsymbol{\mu}[f(\mathbf{Y})] = K(\mathbf{Y},\mathbf{X})\left(K(\mathbf{X},\mathbf{X}) + \mathrm{Cov}_{\mathrm{data}}(\mathbf{X},\mathbf{X})\right)^{-1} f(\mathbf{X}) \tag{9}$$

$$\boldsymbol{\Sigma}[f(\mathbf{Y})] = K(\mathbf{Y},\mathbf{Y}) - K(\mathbf{Y},\mathbf{X})\left(K(\mathbf{X},\mathbf{X}) + \mathrm{Cov}_{\mathrm{data}}(\mathbf{X},\mathbf{X})\right)^{-1} K(\mathbf{X},\mathbf{Y}). \tag{10}$$

This can be computationally expensive thanks to the matrix multiplications and matrix inverses, but it's pretty easy to write down on a computer. Gaussian process regression becomes problematic for large data sets because matrix inversion has a computational cost of n^3.

You might not get much insight from Equations (9) and (10) (I don't, anyway), so let's plug in a few simple examples to see how it works.

Example 1: noise-free recovery of measurements
Suppose that my measurement errors on my data are zero, and that I see what model values I recover at the points $\mathbf{X}$ where I actually have data. I want to check that Gaussian process regression interpolates straight through these points. I will set

$$\mathbf{Y} = \mathbf{X}\mathrm{Cov}_{\mathrm{data}}(\mathbf{X},\mathbf{X}) = 0. \tag{11}$$

I will then plug into Equations (9) and (10):

$$\begin{aligned}
\boldsymbol{\mu}[f(\mathbf{X})] &= K(\mathbf{X},\mathbf{X})\left(K(\mathbf{X},\mathbf{X})\right)^{-1} f(\mathbf{X}) && (12)\\
&= f(\mathbf{X}) && (13)\\
\boldsymbol{\Sigma}[f(\mathbf{X})] &= K(\mathbf{X},\mathbf{X}) - K(\mathbf{X},\mathbf{X})\left(K(\mathbf{X},\mathbf{X})\right)^{-1} K(\mathbf{X},\mathbf{X}) && (14)\\
&= K(\mathbf{X},\mathbf{X}) - K(\mathbf{X},\mathbf{X}) && (15)\\
&= \mathbf{0} && (16)
\end{aligned}$$

So I recover my noise-free measurements. If I am very, very close to the measurement points $\mathbf{X}$, my inferred model values will be close to the measured values and the uncertainties will be small. Again, this is the completely noise-free case.

Example 2: noisy measurements of a function assumed to be constant
Now let's try the case where I have noise that is independent and identically distributed for each measurement. I'll also assume that my covariance matrix for the data is constant. This imposes a requirement that the model be a constant, i.e., have the same value everywhere. Denoting a matrix of all ones $\mathbf{P}$, I will say that

$$K(\mathbf{X},\mathbf{X}) = a^2\mathbf{P} = a^2\begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \dots & \dots & 1 \end{bmatrix}. \tag{17}$$

In this case, fitting a constant model to a bunch of data with uniform error bars is the same as just taking the average of all of the measurements:

$$\langle f \rangle = \left(\sum \frac{x_i}{\sigma^2}\right)\left(\sum \frac{1}{\sigma^2}\right)^{-1} = \frac{1}{n}\sum x_i \tag{18}$$

Let's see what Gaussian process regression gives us. I'll use the result

$$a^2\mathbf{P} + \sigma^2\mathbf{I} = \left(\frac{1}{\sigma^2}\mathbf{I} - \frac{a^2}{\sigma^2}\frac{1}{a^2n+\sigma^2}\mathbf{P}\right)^{-1} \tag{19}$$

which you verify by substitution and using $\mathbf{PP} = n\mathbf{P}$. In this case, Equation (9) becomes

$$\begin{aligned}
\boldsymbol{\mu}[f(\mathbf{X})] &= K(\mathbf{X},\mathbf{X})\left(K(\mathbf{X},\mathbf{X})\right)^{-1} f(\mathbf{X}) && (20)\\
&= a^2\mathbf{P}\left(\frac{1}{\sigma^2}\mathbf{I} - \frac{a^2}{\sigma^2}\frac{1}{a^2n+\sigma^2}\mathbf{P}\right) f(\mathbf{X}) && (21)\\
&= \left(\frac{a^2}{\sigma^2}\mathbf{P} - \frac{a^4n}{\sigma^2}\frac{1}{a^2n+\sigma^2}\mathbf{P}\right) f(\mathbf{X}) && (22)\\
&= \left(\frac{a^2}{a^2n+\sigma^2}\mathbf{P}\right) f(\mathbf{X}) && (23)\\
&= \left(\frac{1}{n+\sigma^2/a^2}\mathbf{P}\right) f(\mathbf{X}). && (24)
\end{aligned}$$

This isn't quite the same as just averaging all of my points together: that would be the where I multiply $f(\mathbf{X})$ by a row of ones, sum up, and divide by n to take the average. This equation gives a result that is a little smaller than the average, due to that prior on the model being zero. In the limit of no measurement error ($\sigma = 0$) or infinite variance on the model ($a \to \infty$) I recover the earlier result. You can plug into the equation for the variance and find that

$$\begin{aligned}
\boldsymbol{\Sigma}[f(\mathbf{X})] &= K(\mathbf{X},\mathbf{X}) - K(\mathbf{X},\mathbf{X})\left(K(\mathbf{X},\mathbf{X})\right)^{-1} K(\mathbf{X},\mathbf{X}) && (25)\\
&= a^2\mathbf{P} - a^2\mathbf{P}\left(\frac{1}{\sigma^2}\mathbf{I} - \frac{a^2}{\sigma^2}\frac{1}{a^2n+\sigma^2}\mathbf{P}\right) a^2\mathbf{P} && (26)\\
&= \left(\frac{\sigma^2}{n+\sigma^2/a^2}\right)\mathbf{P} && (27)
\end{aligned}$$

where I skipped a few steps of algebra. Again, this variance is smaller than the σ^2/n I had before, because my prior on the model is tugging me toward zero.

A more realistic covariance function

Nobody uses a constant covariance function for the simple reason that if they know their model to be a constant, they would be able to apply the ideas from the beginning of the course. If you don't know what your model is, but you need to assume some form for its variation, people typically use one of a few covariance functions. The most common is the squared exponential,

$$K(x_1, x_2) = a^2 \exp\left[-\frac{(x_1-x_2)^2}{2h^2}\right]. \tag{28}$$

This generalizes to more than one parameter (you can take the distance between two points on a plane, for example). You have to be a tiny bit careful–it does not generalize to a sphere, where the squared exponential is not positive definite (I ran into this subtlety one time in my own work). But the squared exponential usually works fine. It has two parameters: h, and a. They have the following interpretations:

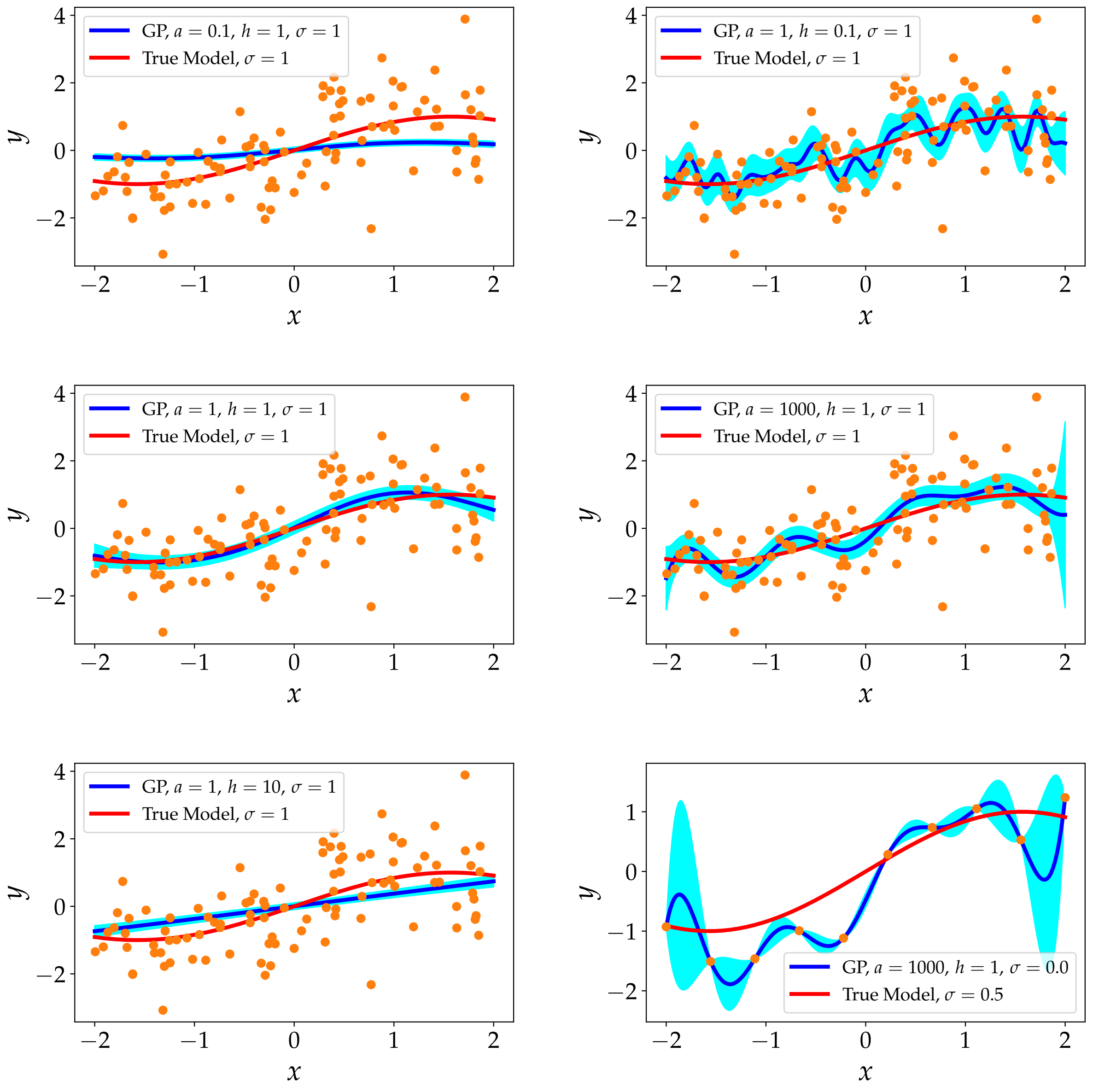
- a tells the Gaussian process how strongly to weight the function's smoothness and tendency towards zero relative to the measurement errors. Recall from the example above, that $a \to \infty$ is equivalent to forcing the model to go right through all of the data points, while $a \to 0$ imposes no meaningful constraints on the model at all.

- h gives a characteristic length scale over which the model varies. There will be a large penalty factor for models that vary over length scales much smaller than h, and almost no penalty for model that vary on scales larger than h.

With this covariance function, we can play around with some sample data to get a bit of intuition. Below, I have plotted some data generated using

```
x = np.random.uniform(-2, 2, 100)
y = np.sin(x) + np.random.randn(len(x))
```

I have then used the square exponential covariance function, Equation (28), varying the parameters a and h. I also add a diagonal matrix to $K(\mathbf{X}, \mathbf{X})$ to account for measurement error in all but one panel. In the last panel, the one with no measurement error, I have used ten equally spaced points. Without any measurement error, using more points results in a very poorly conditioned covariance matrix and numerical problems. In practice, you always want to include measurement error, so this doesn't matter much.



That's a lot of variation in the fit depending on what you choose for the covariance function! If you choose $a \gg \sigma$, the model follows the data too closely, trying to reproduce every wiggle. If you make h too small, the function wiggles a lot over small scales. If you make h too large, it doesn't wiggle enough and you're just fitting a line. If you set a too small ($\ll \sigma$), your prior on the function having a mean of zero becomes more important, and you get tugged to zero (recall the example from earlier). Note that, for the best-fit Gaussian process regression, only the ratio a/σ matters; the actual value of a matters for the covariance matrix of your resulting predictions. The parameters a and h governing your covariance function are called *hyperparameters*. They are parameters that you tune to adjust the nature of the fit.

So what do you do? It's obvious from looking at the images that $a = h = \sigma = 1$ worked better than the alternatives plotted, but how would you figure that out in practice? In other words, how can you go about tuning the hyperparameters for real data? I'll give two answers. One is to write down the probability distribution of the residuals at the location of your training set $\mathbf{X}$. I'll use $f(\mathbf{X})$ to denote my measurements at these points.

$$p(f(\mathbf{X})) = (2\pi)^{-n/2}\sqrt{\det\left(\mathbf{\Sigma}[f(\mathbf{X})]\right)^{-1}}\exp\left[-\frac{1}{2}\left[\boldsymbol{\mu}[f(\mathbf{X}] - f(\mathbf{X})\right]^T\left(\mathbf{\Sigma}[f(\mathbf{X})]\right)^{-1}\left[\boldsymbol{\mu}[f(\mathbf{X}] - f(\mathbf{X})\right]\right] \quad (29)$$

I can then try to adjust the hyperparameters to maximize this probability via a full nonlinear optimization. This approach will usually work all right, but it is prone to falling flat on its face and overfitting anyway and then, how would you know, and how would you fix it?

The second answer, and the one I would recommend, is to use cross-validation. This means to split your data into two or more chunks:

- An initial set for fitting
- A set not used for fitting, but for *cross-validation*
- A set not used in either fitting or cross-validation, to *test* the fidelity of your final model

The last set is less important than the second. The cross-validation set is how you tune the hyperparameters. You save a fraction of your data (10 or 10s of percent), not using them in your Gaussian process regression, or in some other fit of hyperparameters. You then use your Gaussian process regression to predict the values of your cross-validation data, and compare them to the actual values. You can then tune your hyperparameters to get the best agreement.

Cross-validation is a very important step to avoid overfitting. In the example figures above, the Gaussian process regression with $a = 1000$ and $h = 1$ had a standard deviation of the residuals of 0.92. The regression with $a = 1$ and $h = 0.1$ had residuals with a standard deviation of 0.82. The one that looks best by eye, by contrast, had residuals with a standard deviation of 0.96. The other models are *overfit* to the data.

I can tell this overfitting if I save half of the data for cross-validation. In this case, I can compute the standard deviation of the residuals between the predicted values at the cross-validation points, and the actual values. Using all data, recall that my standard deviations of the residuals were 0.82, 0.92, and 0.96 for $a = 1, h = 0.1$, $a = 1000, h = 1$, and $a = h = 1$, respectively. These same values become 1.13, 1.22, and 1.03, respectively, when I measure them for half the data held back for cross-validation. Without cross-validation, I would have

- picked the wrong model (overfit my data); and
- underestimated my errors, compounding the problem.

This sort of thing is more common than you might think. Suppose you have a complicated experiment and you fit a whole bunch of parameters to model the instrument response, the background, etc., etc., plus some physical model parameters that you care about. How do you know that you haven't overfit everything and that you can trust the confidence intervals on the parameters that you care about? If you did overfit everything, then not only are your values and confidence intervals unreliable (you might be able to figure this out using bootstrap resampling, but that approach might be costly), but your values could have been better! Cross-validation can help you avoid overfitting and get the best possible results from your experiment in the case where you are fitting a lot of nuisance parameters.