ChE/BE 163: Introduction to Biomolecular Engineering

Gaussian processes for directed evolution

Justin Bois

Caltech

Fall, 2021

1 Learning from regression

The following happens a lot in science. We vary system parameters **x** and make observations **y**. We perform a regression using some theoretical function $f(x)$, which describes how we expect y to vary with x. We then can have a pretty good idea what we would measure, y_* at some parameter x_* that we didn't actually measure. By making a few measurements, the regression helps us say things more generally, even for parameter values we didn't explicitly perform an experiment for.

1.1 An example: hydrolysis of cellulose

To give a concrete example, let's consider the the breakdown of α-1-methylglucopyranoside, a key step in the hydrolysis of cellulose. This is featured in a nice paper by Wolfenden and Snider about the power of enzymes as catalysts. This example shows that glucoside hydrolysis is incredibly slow in the absence of enzymes.

The chemical rate constant is often well described by the **Arrhenius relation**,

$$k = A\mathrm{e}^{-E_a/k_BT}, \tag{1.1}$$

where E_a is the activation energy that catalysts serve to decrease. For ease of notation, I will define units such that $k_B = 1$, and will convert back to familiar units when needed. So, we can write the Arrhenius rate law as

$$k = A\mathrm{e}^{-E_a/T}. \tag{1.2}$$

In an experiment, we can vary (and exactly measure) T, so there are two parameters, A and E_a. We can find the values of A and E_a that best fit measured data, and the result is shown in Fig. 1

Now, let's look at some data. In Fig. 1, I show the measured chemical rate constant versus temperature. For the best fit parameters, I got that $A \approx 950\ \mathrm{s}^{-1}$ and E_a/k_B a whopping 6000 K. This means that uncatalyzed oxidation is extremely slow.

Importantly, from the regression, we can predict what the chemical rate constant will be at a temperature of 500 K, even though we did not measure it there. It is about $0.005\mathrm{s}^{-1}$. By sampling a few temperatures, we now have some knowledge about the rate constant over a whole range of temperatures.

This is a specific example, but if you want to think more generally, you can think of data sets **y** as a function of **x** parametrized by parameters θ.

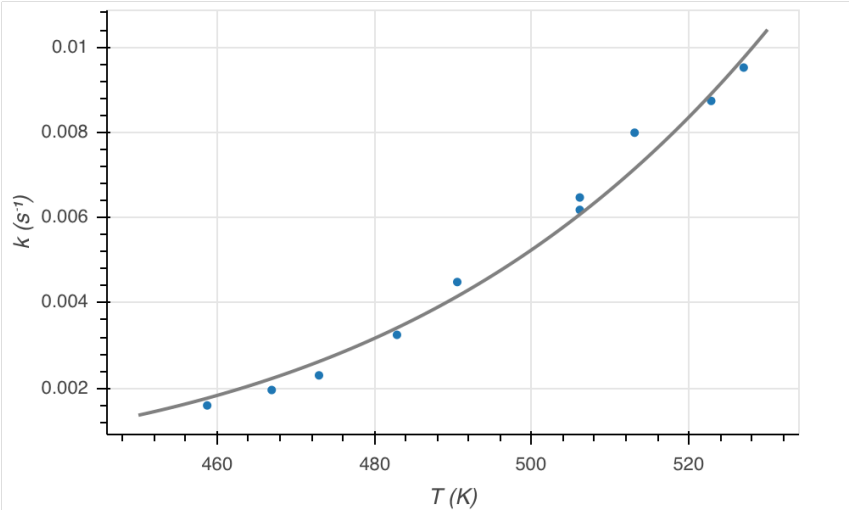

Figure 1: Observed rate constant versus temperature for uncatalyzed hydrolysis of α-1-methylglucopyranoside. Data digitized from Wolfenden and Snider, *Acc. Chem. Res.*, **34**, 938–945, 2001. The best fit Arrhenius rate law is shown in gray.

1.2 Learning from regression in the context of directed evolution

During the course of lots and lots of directed evolution experiments, we have varied sequences and measured fitness of a protein through some screen. It would be great if we could perform a regression so that we could *predict* the fitness of an unobserved sequence. That is our goal in this section of the course. For reasons we have discussed so far, it is almost impossible to do a **parametric** regression as we did for the Arrhenius rate law in the example above. We instead need to resort to **nonparametric** approaches. These approaches do not assume a specific functional form of $f(x)$, but consider an infinite space of possible functions $f(x)$. We will use **Gaussian processes** to do this. However, before proceeding to the nonparametric regression, I will lay some of the theoretical groundwork by demonstrating how a parametric regression is done. We start with Bayes's Theorem.

2 Bayesian probability

Bayes's Theorem may be familiar to you, but let's write it down and think carefully about what it means. We will not define what probability means, or how it is interpreted at the moment. For now, just think of probability as you might intuitively think about it.[1]

[1] I know this sounds weird and totally not rigorous, but I ask you to suspect disbelief.

Let $P(A)$ be the probability of A and $P(B)$ be the probability of B. Further, let $P(A, B)$ be the probability of both A and B. Clearly, $P(A, B) = P(B, A)$, since "and" is commutative.

Now, we define $P(A \mid B)$ as the probability of A *given that* B *is true*. We ofter say this as "the probability of A conditioned on B," and refer to $PA \mid B)$ as a **conditional probability**. Now, it stands to reason that

$$P(A, B) = P(A \mid B)P(B). \tag{2.1}$$

That is to say that the probability of A and B is given by the probability of B times the probability of A, given that B is true.[2] Now, because $P(A, B) = P(B, A)$, we may write

$$P(A, B) = P(A \mid B)P(B) = P(B, A) = P(B \mid A)P(A). \tag{2.2}$$

Rearranging this equation, we arrive at **Bayes's Theorem**.

$$P(A \mid B)P(B) = \frac{P(B \mid A)P(A)}{P(B)}. \tag{2.3}$$

Bayes's Theorem holds for any legitimate interpretation of probability.[3] Two interpretations of probability are most common.

2.1 Interpretations of probability

Frequentist probability. In the *frequentist* interpretation of probability, the probability $P(A)$ represents a long-run frequency over a large number of identical repetitions of an experiment. These repetitions can be, and often are, hypothetical. The event A is restricted to propositions about *random variables*, a quantity that can very meaningfully from experiment to experiment.[4]

Bayesian probability. Here, $P(A)$ is interpreted to directly represent the degree of belief, or plausibility, about A. So, A can be any logical proposition.

We will use the Bayesian interpretation here, and we will indeed apply the notion of probability to logical propositions that are not random variables. Specifically, we will consider probabilities of functions that describe how mutations in protein sequences affect function.

[2] In fact, equation (2.1) often serves to define conditional probability.

[3] We have not formally *defined* probability here, and will only talk about interpretation. You can find a formal definition of probability, for example, on page 20 of Blitzstein and Hwangs excellent Introduction to Probability.

[4] More formally, a random variable transforms the possible outcomes of an experiment to real numbers.

2.2 Bayes's Theorem as a model for learning

Let's go back to our regression of using the Arrhenius relation. Recall that there are two parameters, A and E_a. Prior to doing an experiment, we know some things about these parameters. We know that A is positive, and further that it describes the frequency of properly aligned molecular interactions. Beyond that, we do not know much. So, we could say that before observing data, A could be pretty much anything. Similarly, we do not know much about what E_a is either. It can be positive or negative, and can take most values. We can codify our knowledge of these parameters using probability distributions. Specifically, we might say

$$\log_{10} A \sim \text{Normal}(-6, 6), \qquad A = 10^{\log_{10} A}, \tag{2.4}$$

$$\log_{10} E_a \sim \text{Normal}(-6, 6), \qquad E_a = 10^{\log_{10} E_a}, \tag{2.5}$$

where we have abused notation by taking a logarithm of a quantity with units, but the meaning should be clear. These prior distributions codify our knowledge about these parameters ahead of seeing any data. Specifically, we choose Log-Normal distributions allowing for a wide range of positive values for both A and E_a. I will not explicitly write out the probability distribution $P(A, E_a)$ here, but rather plot it.

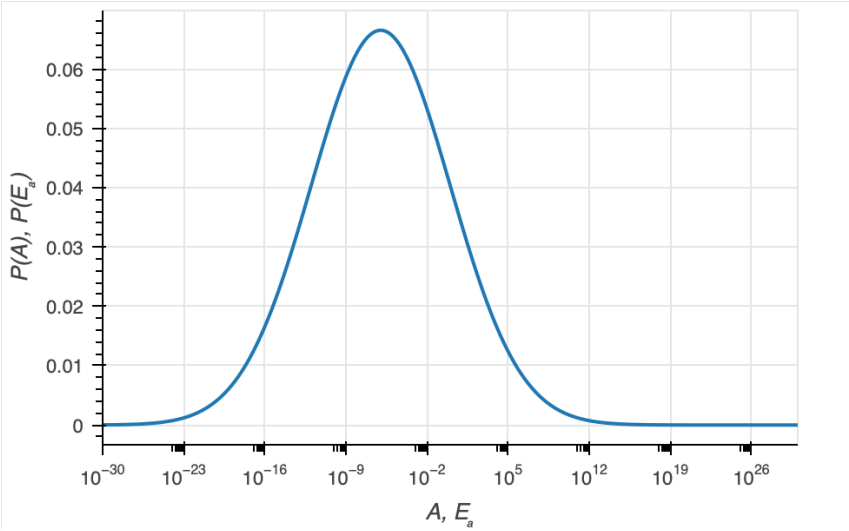

Figure 2: Prior distribution for A in units of s^{-1} and E_a in units of K.

Now, let's look at some data, show in Fig. 1 (ignore the regression line for a moment). How do we think these data were generated? We might assume that there is going to be some error in the measurement of the rate constant, and that this error is Gaussian distributed. In other words,

$$k_{\text{measured}}(T) = A\text{e}^{-E_a/T} + \varepsilon\,, \tag{2.6}$$

$$\varepsilon \sim \text{Normal}(0, \sigma^2), \tag{2.7}$$

where we have specified σ^2 to be the variance describing error in measurement. If we assume that σ^2 is the same for all points, we say that we have **homoscedastic error**. So, we have now also described how the data are generated. Writing out the full expression, we have

$$P(\mathbf{k}, \mathbf{T} \mid A, E_a) = \left(\frac{1}{2\pi\sigma^2}\right)^{n/2} \prod_i \exp\left[-\frac{\left(k_i - A\mathrm{e}^{-E_a/T_i}\right)^2}{2\sigma^2}\right]. \tag{2.8}$$

Here, we have defined $\mathbf{k}$ and $\mathbf{T}$ as the observed data set, which we index over i. We have also assumed each measurement is independent of all others. Equivalently, we may write this as

$$k_i, T_i \mid A, E_a \sim \mathrm{Norm}(A\mathrm{e}^{-E_a/T_i}, \sigma^2). \tag{2.9}$$

Ultimately, we want to learn about the parameters A and E_a *after* seeing the data, so we want to know $P(A, E_a \mid \mathbf{k}, \mathbf{T})$. Here is where Bayes's Theorem comes in! Refering to equation (2.3), we take our logical conjectures A and B to be

$$A \to A, E_a, \tag{2.10}$$

$$B \to \mathbf{k}, \mathbf{T}. \tag{2.11}$$

Then, we have

$$P(A, E_a \mid \mathbf{k}, \mathbf{T}) = \frac{P(\mathbf{k}, \mathbf{T} \mid A, E_a)\, P(A, E_a)}{P(\mathbf{k}, \mathbf{T})}. \tag{2.12}$$

In the generic example of a regression, this is

$$P(\theta \mid \mathbf{y}, \mathbf{x}) = \frac{P(\mathbf{y}, \mathbf{x} \mid \theta)\, P(\theta)}{P(\mathbf{y}, \mathbf{x})}. \tag{2.13}$$

We have specified the two terms in the numerator. The denominator is already determined by the fact that $P(A, E_a \mid \mathbf{k}, \mathbf{T})$ must be normalized.

$$P(\mathbf{k}, \mathbf{T}) = \int \mathrm{d}A \int \mathrm{d}E_a\, P(\mathbf{k}, \mathbf{T} \mid A, E_a)\, P(A, E_a). \tag{2.14}$$

So, we have fully specified what we are after. Bayes's Theorem tells us how we learned from the data. *Prior* to seeing the data, we didn't know much about the parameters, as was clear by $P(A, E_a)$ being Uniform. After seeing the data, or *a posteriori*, we have a new distribution describing our knowledge of the parameters. The connection came through $P(\mathbf{k}, \mathbf{T} \mid A, E_a)$, the *likelihood* of observing our data given the parameters.

We can give the terms in Bayes Theorem convenient names.

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}. \tag{2.15}$$

The prior probability. In our case, this is $P(A, E_a, \sigma)$. This is what we knew about the parameters *prior* to seeing the data. Notice how I have snuck σ into the prior. This is because σ is a parameter, and it is also something we want to learn about.

The likelihood. The likelihood, $P(\mathbf{k}, \mathbf{T} \mid A, E_a, \sigma)$, describes how likely it is to acquire the observed data, *given a set of parameters*.

The marginal likelihood. This is the denominator, $P(\mathbf{k}, \mathbf{T}, \sigma)$. I will not talk much about this here, except to say that it is a normalization constant for the posterior. Note that it does *not* depend on the parameters, and is entirely determined from the likelihood and prior, so we do not need to consider it explicitly in our modeling.

The posterior probability. This is what we are after, $P(A, E_a, \sigma \mid \mathbf{k}, \mathbf{T})$. This codifies our knowledge of the parameters after seeing the data.

So, in summary, we **learn** about the parameters, moving from our prior knowledge to our posterior knowledge via the likelihood.

2.3 The posterior predictive distribution

Let us now formalize how we can use the posterior distribution to *predict* what we would expect from a new measurement of the rate constant, k_* at some temperature we have not yet measured, T_*. We seek the **posterior predictive distribution**, $P(k_* \mid T_*, \mathbf{k}, \mathbf{T})$. This is what we would expect to measure, given that we have already observed some values of k. Generally, we if have a set of parameters θ and a measured data set $\mathbf{x}$, $\mathbf{y}$, then the posterior predictive distribution is computed by marginalizing over the posterior.[5]

$$P(y_* \mid x_*, \mathbf{x}, \mathbf{y}) = \int \mathrm{d}\theta \, P(y_* \mid x_*, \theta) \, P(\theta \mid x_*, \mathbf{x}, \mathbf{y}) \tag{2.16}$$

$$= \int \mathrm{d}\theta \, P(y_* \mid x_*, \theta) \, P(\theta \mid \mathbf{x}, \mathbf{y}).$$

The first term in the integral is the likelihood of observing y_* for a set of parameters, and the second term describes the probability distribution of those parameters, given that we have measured some data. In this case of our present example,

$$P(k_* \mid T_*, \mathbf{k}, \mathbf{T}) = \int \mathrm{d}A \int \mathrm{d}E_a \, P(k_* \mid T_*, A, E_a) \, P(A, E_a \mid \mathbf{kT}). \tag{2.17}$$

[5] In the first line of equation (2.17), I have been explicit in including all terms for proper marginalization, and in the second line noted that the posterior probability distribution of θ has no x_* dependence, so $P(\theta \mid x_*, \mathbf{x}, \mathbf{y}) = P(\theta \mid \mathbf{x}, \mathbf{y})$.

2.4 Summarizing the posterior

We often wish to summarize the posterior with a few compact numbers. One such summary is the **maximum a posteriori probability**, or MAP, which is the value of parameters that maximize the posterior. This is typically what you find when you do a parametric regression, and is computing using optimization methods. I computed this, and I got the results reported above. The result of the most probable curve for the data is shown in Fig. 1.

3 Gaussian processes

When trying to predict protein fitness, we do not have the convenience of an Arrhenius rate law to fit to the data. Having that function was convenient because we know exactly what parameters we needed to describe with probability distributions. We described A and E_a as Uniformly distributed, and we defined the likelihood to be Gaussian, with a mean given by what would be predicted by the Arrhenius rate law. But, alas, we have no such function for protein fitness as a function of sequence. Actually, we will need to define a distance between sequences, and we will use a sequence-structure distance, which we define later. Nonetheless, we have no function relating the sequence-structure distance to fitness.

3.1 Processes and nonparametric Bayes

Remember that Bayes's Theorem applies to any logical conjecture. It even applies to *functions*! So, imagine we have observed data $\mathbf{X}$ and y. I use a capital $\mathbf{X}$ here to allow for multidimensional dependent variables. For example, we might want to study both temperature and pH dependence of a rate constant. In this case, each row of $\mathbf{X}$ is a pH, temperature pair. We define row i of $\mathbf{X}$ to be $\mathbf{x}_i$.

We expect that for each observation, $y_i = f(\mathbf{x}_i) + \varepsilon_i$, where ε_i is some measurement error and $f(x)$ is an *unknown* function of $\mathbf{x}$. We can still write Bayes's Theorem.

$$P(f \mid \mathbf{y}, \mathbf{X}) = \frac{P(\mathbf{y}, \mathbf{X} \mid f)\, P(f)}{P(\mathbf{y}, \mathbf{X})}. \tag{3.1}$$

This may seem strange to write a probability of functions. We call a probability distribution over functions a **process**. So, the posterior, $P(f \mid \mathbf{y}, \mathbf{X})$, and the prior, $P(f)$, are processes.

Assuming for a moment we can compute the posterior, we then then again come up with a posterior predictive distribution,

$$P(f(\mathbf{x}_*) \mid \mathbf{x}_*, \mathbf{y}, \mathbf{X}) = \int \mathrm{d}f\, P(f(\mathbf{x}_*) \mid \mathbf{x}_*, f)\, P(f \mid \mathbf{X}, \mathbf{y}). \tag{3.2}$$

This looks exactly the same as for parametric regression. Instead of writing a probability distribution over infinitely many parameter values in the parametric setting, we are no writing a process over infinitely many functions. This posterior predictive distribution is key for us. We want to be able to *predict* what a new protein fitness will be for an unobserved sequence.

3.2 Gaussian processes with finite points

How can we treat a probability distribution over functions? We can instead define a probability distribution over the function's *values* at some arbitrary points. So, imagine we measure a function at points $\mathbf{X}$, with N total observations. We can define a **joint distribution**, $P(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N))$. We use this probability density as a drop-in replacement for a process.

So far, we have said nothing about what the joint distribution is. We can choose many distributions for this, but we might choose a joint Gaussian distribution. This defines a **Gaussian process**, or GP. To define the joint distribution, then, we need to define the two parameters of a multivariate Gaussian distribution, its mean and covariance. The mean must be defined for an arbitrary point $\mathbf{x}$, and the covariance for an arbitrary pair of points $\mathbf{x}, \mathbf{x}'$. Thus, the mean function, $m(\mathbf{x})$ and the covariance function, $k(\mathbf{x}, \mathbf{x}')$, uniquely define a Gaussian process. We can write a Gaussian process as

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')). \tag{3.3}$$

Note that because in practice, we compute $f(\mathbf{x}_*)$, that is the function we are fitting are a finite set of points $\mathbf{x}_*$ we can write Bayes's theorem again as

$$P(f(\mathbf{x}_*) \mid \mathbf{x}_*, \mathbf{y}, \mathbf{X}) = \frac{P(\mathbf{y}, \mathbf{X} \mid f(\mathbf{x}_*), \mathbf{x}_*)\, P(f(\mathbf{x}_*) \mid \mathbf{x}_*)}{P(\mathbf{y} \mid \mathbf{X})}. \tag{3.4}$$

Or, if we want to evaluate f at many points, $\mathbf{X}_*$,

$$P(\mathbf{f}(\mathbf{X}_*) \mid \mathbf{X}_*, \mathbf{y}, \mathbf{X}) = \frac{P(\mathbf{y}, \mathbf{X} \mid \mathbf{f}(\mathbf{X}_*), \mathbf{X}_*)\, P(\mathbf{f}(\mathbf{X}_*) \mid \mathbf{X}_*)}{P(\mathbf{y} \mid \mathbf{X})}. \tag{3.5}$$

So, the posterior distribution is a predictive distribution.

3.3 The mean function and centering and scaling

In a purely nonparametric approach, we almost always take the mean function to be zero; $m(\mathbf{x}) = 0$. We may, however, with to do a **semi-parametric regression** and

introduce an explicit bias via $m(\mathbf{x})$. This is interesting, but outside of our present discussion. Henceforth, in this class, we will take $m(\mathbf{x}) = 0$.

In general in machine learning applications, and many nonparametric contexts in general, it is good practice to **center** and **scale** the observed data to improve performance of your learning algorithms. I will not get into the details of this here[6], but rather will encourage you to center and scale your data before fitting a GP. Specifically, if $\bar{\mathbf{x}}$ is the arithmetic mean of observations $\mathbf{x}$, and $s_{\mathbf{x}}$ is the sample standard deviation of $\mathbf{x}$, then you should apply a linear transformation of $\mathbf{x}$ to get a scaled version.

$$\mathbf{x}_{\text{scaled}} = \frac{\mathbf{x} - \bar{\mathbf{x}}}{s_{\mathbf{x}}}. \tag{3.6}$$

You should then work with $\mathbf{x}_{\text{scaled}}$. You should do this to all $\mathbf{x}$ and $\mathbf{y}$ values. You can then apply the inverse linear transformation to get back your original values after fitting the GP.

$$\mathbf{x} = s_{\mathbf{x}}\mathbf{x}_{\text{scaled}} + \bar{\mathbf{x}}. \tag{3.7}$$

3.4 The kernel and covariance matrix

The covariance function, $k(\mathbf{x}, \mathbf{x}')$ is called a **kernel**. It must have certain properties. Remember that it defined the entries of a covariance matrix of a multivariate Gaussian distribution. Specifically, let's say that $\mathbf{X}$ has n rows. Then, we can define an $n \times n$ matrix, $\mathsf{K}(\mathbf{X}, \mathbf{X}')$ that has entries

$$K_{ij} = k(\mathbf{x}_i, \mathbf{x}'_j). \tag{3.8}$$

This matrix is called a **covariance matrix**, which is a special case of a **Gram matrix**. Because this is a joint Gaussian distribution, the covariance matrix $\mathsf{K}(\mathbf{X}, \mathbf{X}')$ must be positive definite for *any* $\mathbf{X}$, $\mathbf{X}'$. This puts a restriction on what kernels that are allowed. Some common kernels are shown below.

$$\text{polynomial: } k(\mathbf{x}, \mathbf{x}') = (\sigma_0^2 + \sigma_p^2 \mathbf{x}^\mathsf{T}\mathbf{x}')^d, \quad d \in \{1, 2, 3, \ldots\} \tag{3.9}$$

$$\text{squared exponential (SE): } k(\mathbf{x}, \mathbf{x}') = \alpha^2 \exp\left[-\frac{\|\mathbf{x} - \mathbf{x}'\|_2^2}{2\rho^2}\right] \tag{3.10}$$

$$\text{Matérn: } k(\mathbf{x}, \mathbf{x}') = \alpha^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \beta^\nu K_\nu(\beta), \text{ where } \beta = \left(\frac{2\nu \|\mathbf{x} - \mathbf{x}'\|_2^2}{\rho^2}\right)^{\frac{1}{2}} \tag{3.11}$$

[6] See the series of blog posts on preprocessing in data science by Hugo-Bowne Anderson for an accessible discussion on why centering and scaling is important.

$$\text{Matérn } (\nu = 5/2)\text{: } k(\mathbf{x}, \mathbf{x}') = \alpha^2 \left(1 + \left(\frac{5\,\|\mathbf{x} - \mathbf{x}'\|_2^2}{\rho^2}\right)^{\frac{1}{2}} + \frac{5\,\|\mathbf{x} - \mathbf{x}'\|_2^2}{3\rho^2}\right) \times \exp\left[-\left(\frac{5\,\|\mathbf{x} - \mathbf{x}'\|_2^2}{\rho^2}\right)^{\frac{1}{2}}\right], \tag{3.12}$$

where

$$\|\mathbf{x} - \mathbf{x}'\|_2^2 = (\mathbf{x} - \mathbf{x}')^\mathsf{T}(\mathbf{x} - \mathbf{x}') \tag{3.13}$$

is the 2-norm, and K_ν is a modified Bessel function of the second kind. All of these kernels are positive definite.

So, how do we interpret all of this? For ease of discussion, let's talk specifically about the SE kernel. If $\mathbf{x}$ and $\mathbf{x}'$ are close to each other, the kernel returns a large value. The value returned by the kernel falls off as $\mathbf{x}$ and $\mathbf{x}'$ grow farther apart. So, the covariance between $\mathbf{x}$ and $\mathbf{x}'$ is large if they are close, and small if they are farther apart. A large covariance means that $f(\mathbf{x})$ and $f(\mathbf{x}')$ should be close to one another, and a small covariance means that they are unrelated.

Finally, we note that each of the kernels have parameters. In the case of the SE kernel, there are two parameters, α and ρ. So, in this sense, this "nonparametric" model has some tunable parameters. Specifically, these parameters say something about how the possible functions $f(\mathbf{x})$ might behave. The covariance is modulated by ρ. If ρ is large, then $\mathbf{x}$ and $\mathbf{x}'$ do not have to be so close together to influence each other. This means that the function is not so rapidly varying. So, ρ sets a length scale over which the function $f(\mathbf{x})$ varies. Similarly, α sets the amplitude of the variations. The larger α is, the more $f(\mathbf{x})$ will vary in the vertical direction. The parameters of the kernel are often called **hyperparameters**.

In summary, the kernel specifies key features of the functions we are using to describe our data. It sets lengths scales for typical variation in the horizontal and vertical directions.

3.5 The prior

We can visualize the effects of the parameters by **sampling** out of the prior distribution. Remember that we can represent the prior $P(f \mid \alpha, \rho)$ as a multivariate Gaussian distribution over a set of finite points. For ease, let's consider a one-dimensional dependent variable, so $\mathbf{x} = x$. Saw we want to evaluate $f(x)$ at a set of positions $\mathbf{x}_*$. Then,

$$f(\mathbf{x}_*) \mid \alpha, \rho \sim \text{Norm}(0, K(\mathbf{x}_*, \mathbf{x}_*)). \tag{3.14}$$

To sample out of this distribution, we compute the matrix $K(\mathbf{x}_*, \mathbf{x}_*)$ for given values of α and ρ, and then we sample out of this distribution, e.g., using `np.random.multivariate_normal()`. In Fig. 3, I show samples out of a prior with a SE kernel with $\alpha = 1$ and various values of ρ. I have plotted the results as dots to emphasize that we are evaluating $f(\mathbf{x}_*)$ at a finite set of points. We see that as ρ grows, the undulations decrease in frequency.

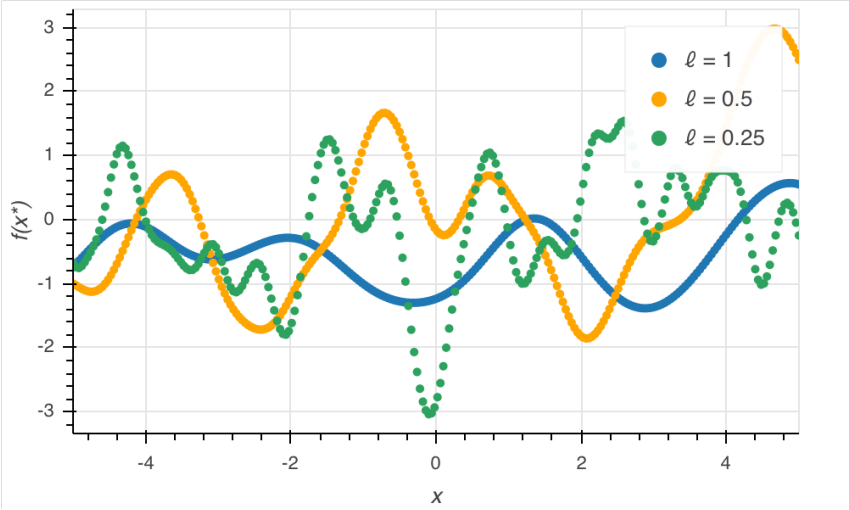

Figure 3: Priors for a SE kernel with $\alpha = 1$ and various values of ρ.

We can draw many many functions this way. The result is shown in Fig. 4. The function mostly lies between -2 and 2, which should contain about 95% of the function values since $\alpha = 1$. As you can see, prior to obtaining data, the function we are after can be anything varying on a length scale of order one unit with an amplitude of order one unit.

3.6 The likelihood

We expect our observation y to follow $y = f(\mathbf{x})$, perhaps with some error. We can model that error to be Normally distributed with homoscedastic error σ, i.e.,

$$y_i \sim \text{Norm}(f(\mathbf{x}_i), \sigma) \quad \forall i. \tag{3.15}$$

This choice of likelihood has the important consequence that it is the **conjugate** likelihood to a Gaussian process prior. This means that the posterior distribution of $f(\mathbf{x})$ is also a Gaussian process. I.e.,

$$f(\mathbf{x}) \mid \mathbf{y}, \mathbf{X} \sim GP(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}' + \delta_{\mathbf{x},\mathbf{x}'}\sigma)). \tag{3.16}$$

Here, I show the posterior explicitly, which can be calculated analytically for the Gaussian likelihood and GP prior. The posterior has the same mean function as the

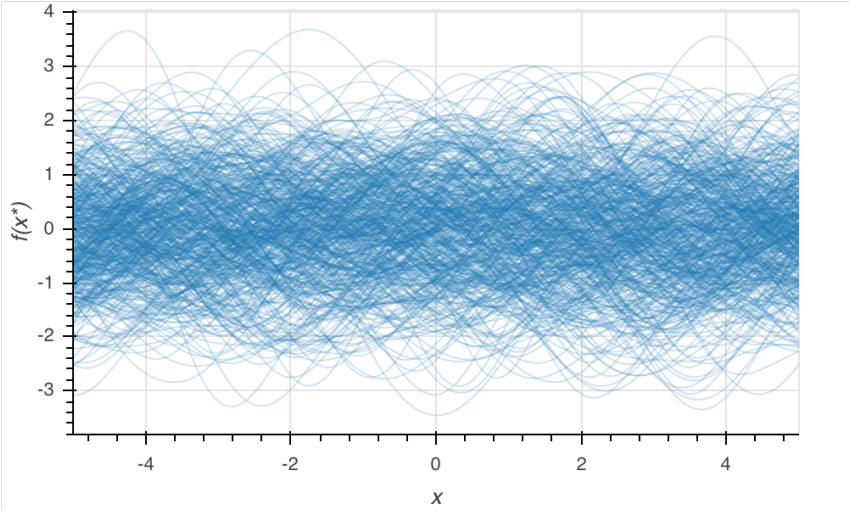

Figure 4: A plot of 500 functions drawn from a GP prior using a SE kernel with $\alpha = \rho = 1$.

prior, and the kernel is adjusted by adding σ whenever $\mathbf{x} = \mathbf{x}'$, as denoted with

$$\delta_{\mathbf{x},\mathbf{x}'} = \begin{cases} 1 & \mathbf{x} = \mathbf{x}' \\ 0 & \mathbf{x} \neq \mathbf{x}'. \end{cases} \tag{3.17}$$

So, for many observations $\mathbf{X}$ with a zero mean function, we define

$$\mathsf{K}_\mathbf{y} \equiv \mathsf{K}(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathsf{I}, \tag{3.18}$$

where I is the identity matrix. Then, we have

$$\mathbf{f}(\mathbf{x}) \sim \text{Norm}(0, \mathsf{K}_\mathbf{y}). \tag{3.19}$$

3.7 The posterior predictive distribution

We can find the posterior predictive distribution be integrating the posterior, but, there is clever way to get to the distribution directly using properties of multivariate Gaussian distributions.

We can specify a *joint* distribution of the posterior of the observed points, $\mathbf{f}(\mathbf{X})$ and the desired set of function evaluations, $\mathbf{f}(\mathbf{X}_*)$. We can do this because we know the posterior is a Gaussian process. The joint distribution is again a multivariate Gaussian. For notational convenience, we will define $\mathbf{f} = \mathbf{f}(\mathbf{X})$ to be the function evaluated at measured $\mathbf{X}$, and $\mathbf{f}_* = \mathbf{f}(\mathbf{X}_*)$ to be the function evaluated at unmeasured $\mathbf{X}_*$

$$\begin{pmatrix} \mathbf{f} \\ \mathbf{f}_* \end{pmatrix} \sim \text{Norm}\left(0, \begin{pmatrix} \mathsf{K}_\mathbf{y} & \mathsf{K}_* \\ \mathsf{K}_*^\mathsf{T} & \mathsf{K}_{**} \end{pmatrix}\right). \tag{3.20}$$

For further notational simplicity, we have also defined

$$\mathsf{K}_* = \mathsf{K}(\mathbf{X}, \mathbf{X}_*) \tag{3.21}$$

$$\mathsf{K}_{**} = \mathsf{K}(\mathbf{X}_*, \mathbf{X}_*). \tag{3.22}$$

It is a property of Gaussian matrices (which may be derived with lots of grunge) that we can get the conditional distribution from the joint distribution. The conditional distribution is exactly the posterior predictive distribution. Using this result, we have

$$\mathbf{f}_* \mid \mathbf{X}_*, \mathbf{X}, \mathbf{y} \sim \mathrm{Norm}(\mathbf{m}_*, \Sigma_*), \tag{3.23}$$

where

$$\mathbf{m}_* = \mathsf{K}_*^{\mathsf{T}} \mathsf{K}_{\mathbf{y}}^{-1} \mathbf{y}, \tag{3.24}$$

$$\Sigma_* = \mathsf{K}_{**} - \mathsf{K}_*^{\mathsf{T}} \mathsf{K}_{\mathbf{y}}^{-1} \mathsf{K}_*. \tag{3.25}$$

The vector $\mathbf{m}_*$ gives the mean (and therefore most probable, since we are dealing with Gaussian distributions) value of $\mathbf{f}(\mathbf{X}_*)$. The diagonal entries of Σ_* give the variance, and therefore the uncertainty, in $\mathbf{f}(\mathbf{X}_*)$. Computing $\mathbf{m}_*$ and Σ_* (or at least its diagonal entries) is called **fitting** the Gaussian process. And we now have our predictions!

3.8 Computing the fit

Setting aside the calculation of the posterior covariance matrix Σ_* for the moment, we will focus on the calculation of $\mathbf{m}_*$. At the center of fitting a Gaussian process is inverting the matrix $\mathsf{K}_{\mathbf{y}}$. More specifically, we need to compute $\mathsf{K}_{\mathbf{y}}^{-1}\mathbf{y}$, which is the solution ξ to

$$\mathsf{K}_{\mathbf{y}}\, \xi = \mathbf{y}. \tag{3.26}$$

Because $\mathsf{K}_{\mathbf{y}}$ is symmetric and positive definite by construction[7], it has a **Cholesky decomposition**, L, such that

$$\mathsf{K}_{\mathbf{y}} = \mathsf{L}\mathsf{L}^{\mathsf{T}}. \tag{3.27}$$

The matrix L is lower triangular. Importantly, it can be stably computed in order n^3 operations (for an $n \times n$ $\mathsf{K}_{\mathbf{y}}$). Thus,

$$\mathsf{L}\mathsf{L}^{\mathsf{T}}\, \xi = \mathbf{y}. \tag{3.28}$$

[7] In pracrtice, it is sometimes necessary to add a small constant to each term in the diagonal of $\mathsf{K}_{\mathbf{y}}$ to ensure that numerical rounding errors do not destroy positive definiteness.

Let $\mathbf{z} = \mathsf{L}^\mathsf{T}\xi$. Then we can solve for $\mathbf{z}$ by solving

$$\mathsf{L}\mathbf{z} = \mathbf{y}. \tag{3.29}$$

This is a triangular system and is easily solved. Given $\mathbf{z}$, we then solve for ξ by solving another triangular system

$$\mathsf{L}^\mathsf{T}\xi = \mathbf{z}. \tag{3.30}$$

So, we never directly invert the $\mathsf{K}_\mathbf{y}$ matrix. Given that we can compute ξ, we then directly compute $\mathbf{m}_* = \mathsf{K}_*^\mathsf{T}\xi$.

In practice, we rarely compute the off-diagonal entries of Σ, and only compute the diagonal elements. The diagonal elements give the uncertainty in each predicted $f(\mathbf{x}_*)$. To compute the variance in a single $f(\mathbf{x}_*)$, we define

$$\mathbf{k}_* = (k(\mathbf{x}_1, \mathbf{x}_*), k(\mathbf{x}_2, \mathbf{x}_*), \ldots, k(\mathbf{x}_n, \mathbf{x}_*),)^\mathsf{T} \tag{3.31}$$

to be an array of covariances between $\mathbf{x}_*$ and all of the points $\mathbf{x}$ where measurements were made. Then, the variance is

$$\sigma_*^2 = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\mathsf{T}\mathsf{K}_\mathbf{y}^{-1}\mathbf{k}_*. \tag{3.32}$$

We can use the Cholesky decomposition of $\mathsf{K}_\mathbf{y}$ to compute this as well.

4 Hyperparameters

I have just described how to fit a Gaussian process in order to make predictions. Notice, though, that we did this for a specific set of values of the hypoerparameters. In the case of the SE kernel, those hyperparameters are α and ρ, with the homoscedastic error σ also being a hyperparameter.

We might choose $\alpha = \rho = \sigma = 1$ in for a centered and scaled fit. This makes sense because centering and scaling aims to bring the variation in the data toward unity. Let's do this for the kinetic data. The result is shown in Fig. 5. As is traditionally done, the mean curve of $\mathbf{f}(\mathbf{X}_*)$ is shown as a line, and the shaded region shows plot and minus 1.96 standard deviations to give a 95% credible region.

This fit does not quite hold muster. It tends to dampen out the variation in the data in favor of a flatter curve. Furthermore, the credible region seems unreasonably wide for these data. Maybe this was not the best choice of hyperparameters.

4.1 Hyperparameters in the posterior calculation

To think about how to deal with this, let's go back to Bayes's theorem, equation (3.5), but this time explicitly include the hyperparameters from the SE kernel and

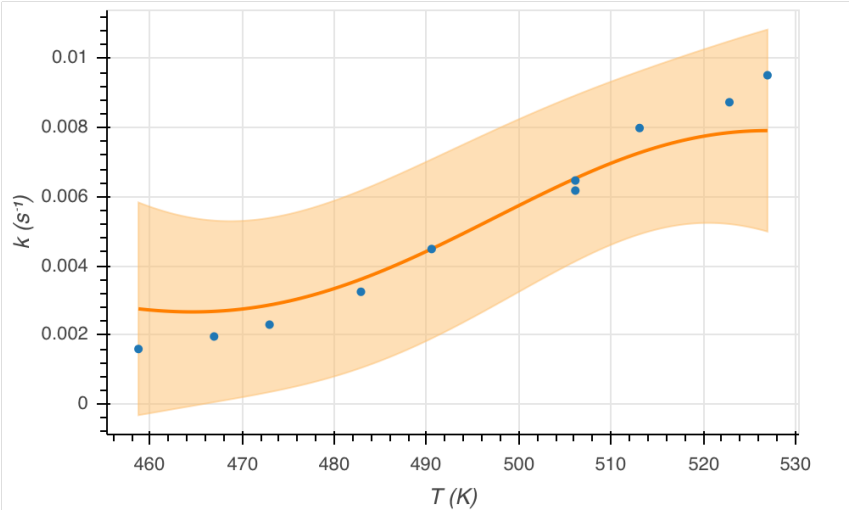

Figure 5: Fit of a Gaussian process to the rate constant data. The hyperparameters for the fit are $\alpha = \rho = \sigma = 1$.

homoscedastic error, which we should have been doing in the first place (but I intentionally didn't so as not to clutter the presentation).

$$P(\mathbf{f}_*, \sigma, \alpha, \rho \mid \mathbf{X}_*, \mathbf{y}, \mathbf{X}) = \frac{P(\mathbf{y}, \mid \mathbf{X}, \mathbf{f}_*, \sigma, \alpha, \rho, \mathbf{X}_*)\, P(\mathbf{f}_*, \sigma, \alpha, \rho \mid \mathbf{X}, \mathbf{X}_*)}{P(\mathbf{y} \mid \mathbf{X}, \mathbf{X}_*)}$$

$$= \frac{P(\mathbf{y} \mid \mathbf{X}, \sigma, \alpha, \rho)\, P(\mathbf{f}_*, \sigma, \alpha, \rho \mid \mathbf{X}_*)}{P(\mathbf{y} \mid \mathbf{X})}. \tag{4.1}$$

In the second line, I have removed variables that are not directly conditioning to reduce clutter. Note that we have explicitly denoted that σ, α, and ρ are unknown. We have also noted that the likelihood does not explicitly depend on them. We can rewrite the prior by converting it from a joint distribution to a conditional one.

$$P(\mathbf{f}_*, \sigma, \alpha, \rho \mid \mathbf{X}_*) = P(\mathbf{f}_* \mid \sigma, \alpha, \rho, \mathbf{X}_*)\, P(\sigma, \alpha, \rho \mid \mathbf{X}_*)$$

$$= P(\mathbf{f}_* \mid \sigma, \alpha, \rho, \mathbf{X}_*)\, P(\sigma, \alpha, \rho). \tag{4.2}$$

Thus, we have

$$P(\mathbf{f}_*, \sigma, \alpha, \rho \mid \mathbf{X}_*, \mathbf{y}, \mathbf{X}) = \frac{P(\mathbf{y} \mid \mathbf{X}, \sigma, \alpha, \rho)\, P(\mathbf{f}_* \mid \sigma, \alpha, \rho, \mathbf{X}_*)\, P(\sigma, \alpha, \rho)}{P(\mathbf{y} \mid \mathbf{X})}. \tag{4.3}$$

We see that this defined a **hierarchical model**; hence the name hyperparameters.

To most rigorously obtain our estimates for $\mathbf{f}_*$, specifically its posterior, we would also need to include σ, α, and ρ in the full Bayesian treatment. This means also

specifying their priors, which would require some thought. We can then marginalize out the hyperpriors to get our desired probability distribution, specifically

$$P(\mathbf{f}_* \mid \mathbf{X}_*, \mathbf{y}, \mathbf{X}) = \int \mathrm{d}\sigma \int \mathrm{d}\alpha \int \mathrm{d}\rho \, P(\mathbf{f}_*, \sigma, \alpha, \rho \mid \mathbf{X}_*, \mathbf{y}, \mathbf{X}). \tag{4.4}$$

This can be achieved with Markov chain Monte Carlo. This is unfortunately computationally intensive, and we might seek a more tractable means.

4.2 Choosing a single optimal set of hyperparameters

A more computationally tractable approach is to attempt to choose a single optimal set of hyperparameters. The technique we lay out here is widely used, and is used in the papers we will read about Gaussian processes being used to predict protein fitness based on a sequence-structure metric. But, I warn you, this approach is fraught with danger. You can pathologically overfit your data, among other problems. A full Bayesian treatment using MCMC is often preferred. For excellent discussion on this, see this series of blog posts from Mike Betancourt. Nonetheless, we proceed.

Our strategy is to find the hyperparameters that maximize the likelihood $P(\mathbf{y} \mid \mathbf{X}, \mathbf{f}, \sigma, \alpha, \rho)$, in effect doing a maximum likelihood estimation of them. We will not directly maximize the likelihood, but rather will marginalize out the effect of the Gaussian process,

$$P(\mathbf{y} \mid \mathbf{X}, \sigma, \alpha, \rho) = \int \mathrm{d}\mathbf{f} \, P(\mathbf{y} \mid \mathbf{X}, \mathbf{f}, \sigma, \alpha, \rho) \, P(\mathbf{f} \mid \mathbf{X}, \sigma, \alpha, \rho), \tag{4.5}$$

which gives the **marginal likelihood**. I will not derive it here, but we can show that

$$\ln P(\mathbf{y} \mid \mathbf{X}, \sigma, \alpha, \rho) = -\frac{1}{2}\mathbf{y}^\mathsf{T}\mathsf{K}_\mathbf{y}^{-1}\mathbf{y} - \frac{1}{2}\ln|\mathbf{K}_\mathbf{y}| - \frac{n}{2}\ln(2\pi). \tag{4.6}$$

Maximizing the log of the marginal likelihood is the same as optimizing the marginal likelihood, so we can find the values of σ, α, and ρ that maximize the above function.

In looking at the two terms (the last one is a constant and immaterial), the first describes how well the GP fits the data and the second describes model complexity (related to an Occam factor). This second term is important for mitigating overfitting (but does not always do the job).

Performing the optimization amounts to writing a function to compute the matrix $\mathsf{K}_\mathbf{y}$ as a function fo the hyperparameters (and the measured $\mathbf{X}$), and then using this, along with the measured data $\mathbf{y}$, to compute the log marginal likelihood. You can then use optimization algorithms, such as BFGS or Powell's method, to find the maximum.

I did this for the rate constant data, and then used these optimal hyperparameters to fit the GP. I got the result in Fig. 6. This fit passes "the eye test" and we may feel comfortable using it for predictions.

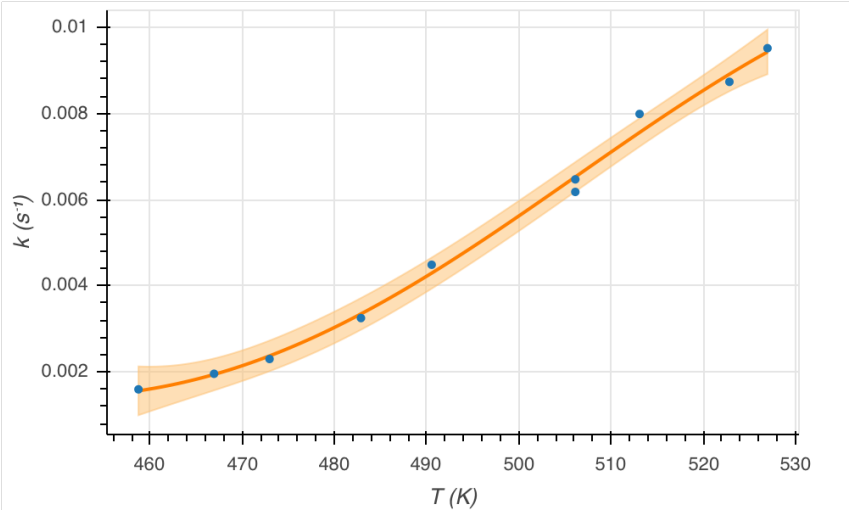

Figure 6: Fit of a Gaussian process to the rate constant data. The hyperparameters for the fit were found using maximum marginal likelihood to be $\sigma = 0.13$, $\rho = 2.54$, and $\alpha = 2.39$.

5 Using GPs to learn protein fitness

In order to apply this methodology to protein fitness, we need to devise a scheme for determining the distance between two proteins, which is $\mathbf{x}$ in all of the analysis above. We can start by defining a sequence distance using one-hot encoding. That is, we set up a binary feature vector that has 20^N entries, where N is the length of the protein. Entry $20i + j$ is one if amino acid j is present at residue i and zero otherwise, where we are using zero indexing. Let $\mathbf{x}_{\text{se}}$ be this sequence vector. The distance between two sequences is then $\|\mathbf{x}_{\text{se}} - \mathbf{x}'_{\text{se}}\|_2 = \sqrt{(\mathbf{x}_{\text{se}} - \mathbf{x}'_{\text{se}})^{\mathsf{T}}(\mathbf{x}_{\text{se}} - \mathbf{x}'_{\text{se}})}$.

We can similarly use one-hot encoding to code for structure similarity. Say that the structure of the parent has N_c contacts, where a contact is defined with the usual 4.5 Å metric we have been using. We then construct an array $\mathbf{x}_{\text{st}}$ with N_c entries, each of which is one for a specific indexed intact contact and zero for a broken contact.

We can then concatenate these two arrays to get our distance measure, $\mathbf{x} = \mathbf{x}_{\text{se}} \frown \mathbf{x}_{\text{st}}$. Note that we could chose to weight entries in either the sequence of structure vector depending on additional knowledge about the protein, but binary feature vectors are easiest to construct and use.

6 Machine learning $\neq$ human learning

You now have the pieces you need to encode and fit a Gaussian process for protein fitness. This is can lead to more effective strategies for chimera construction, as demonstrated in the Bedbrook, Yang, et al. paper. However, it is important to note that machine learning is not the same as human learning. To understand what I mean here, consider the example of fitting a GP regression to the kinetic data we have performed in these notes. The GP regression is *nonparametric*; there is no theoretical model behind it. It is limited in its predictive power to regions near where data were collected. This means that it is not general; it is limited entirely by what data are collected. While the machine has learned, the human has learned little about the mechanisms behind temperature-dependence of chemical rate constants.

Yes, the Arrhenius rate law is phenomenological, but it does have some connection to statistical physics in its similarity to the Eyring equation. Importantly, it has a closed form expression with easy-to-understand components. It can be used to predict reaction rate constants at temperatures beyond those sampled (though there may be some temperature effects on the activation energy and frequency factor).