Covariance Functions

For the most part, our discussion of covariance functions to date has been limited to the commonly-used squared exponential kernel,

$$k(\mathbf{x}, \mathbf{x}') = \lambda^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|_2^2}{2\ell^2}\right).$$

While this covariance function is powerful, it makes some rather strong assumptions about the function being modeled: formally, the squared exponential function is appropriate for modeling functions that are infinitely differentiable. This can most easily be seen through drawing samples from a GP with a squared exponential kernel: Because of the (frequently unrealistic) assumption

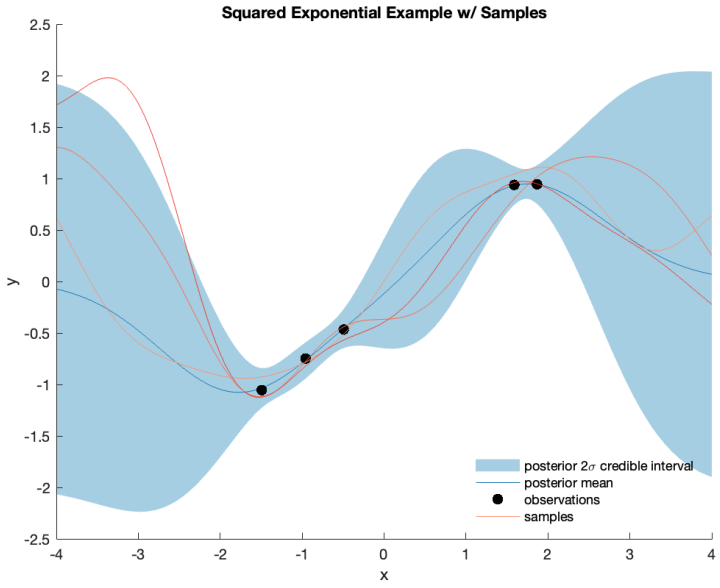


Figure 1: An example of a (zero-mean) Gaussian process with a squared exponential kernel fit to 5 observations of some unknown function as well as three sample paths drawn from the posterior Gaussian posterior. The kernel hyperparameters are set to be $\ell = \lambda = 1$.

of such a high degree of smoothness, many experts have argued against the use of the squared exponential kernel. Here, we will consider a few alternatives and discuss their properties.

The Matérn Covariance Function

The Matérn class of covariance functions is a family of kernels parametrized by a value ν, which controls how smooth functions modeled by the corresponding covariance function are: formally, functions modeled by a Matérn kernel are k-times differentiable if and only if $k < \nu$.

The general functional form is somewhat complex and rarely seen in practice. Instead, a few members of this class corresponding to specific choices of the parameter ν are commonly used:

- When $\nu = \frac{1}{2}$,

$$k(\mathbf{x}, \mathbf{x}') = \lambda^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|_1}{\ell}\right)$$

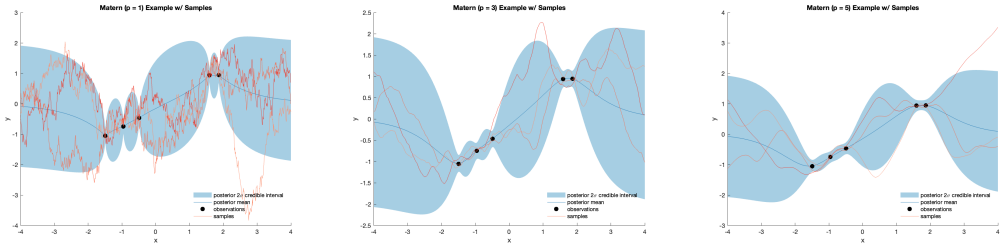
- When $\nu = \frac{3}{2}$,

$$k(\mathbf{x}, \mathbf{x}') = \lambda^2 \left(1 + \sqrt{3}\frac{\|\mathbf{x} - \mathbf{x}'\|_1}{\ell}\right) \exp\left(-\sqrt{3}\frac{\|\mathbf{x} - \mathbf{x}'\|_1}{\ell}\right)$$

- When $\nu = \frac{5}{2}$,

$$k(\mathbf{x}, \mathbf{x}') = \lambda^2 \left(1 + \sqrt{5}\frac{\|\mathbf{x} - \mathbf{x}'\|_1}{\ell} + 5\frac{\|\mathbf{x} - \mathbf{x}'\|_2^2}{3\ell^2}\right) \exp\left(-\sqrt{5}\frac{\|\mathbf{x} - \mathbf{x}'\|_1}{\ell}\right)$$

Figure 2 shows examples of a GP prior with these covariance functions conditioned on the same set of observations; note how the samples from these GPs start out quite jagged but become smoother as ν increases. Indeed, for larger values of ν, the modeled functions becomes so smooth as to be basically indistinguishable from those modeled by a squared exponential kernel. In fact, as $\nu \to \infty$ approaches infinity, the corresponding Matérn covariance function precisely approaches the squared exponential kernel.



(a) $\nu = \frac{1}{2}$ (b) $\nu = \frac{3}{2}$ (c) $\nu = \frac{5}{2}$

Figure 2: Gaussian process posteriors with Matérn covariance functions conditioned on the same set of 5 observations as well as three sample paths from each posterior. Each Matérn kernel uses $\ell = \lambda = 1$ and a different value of the hyperparameter ν.

The Rational Quadratic Covariance Function

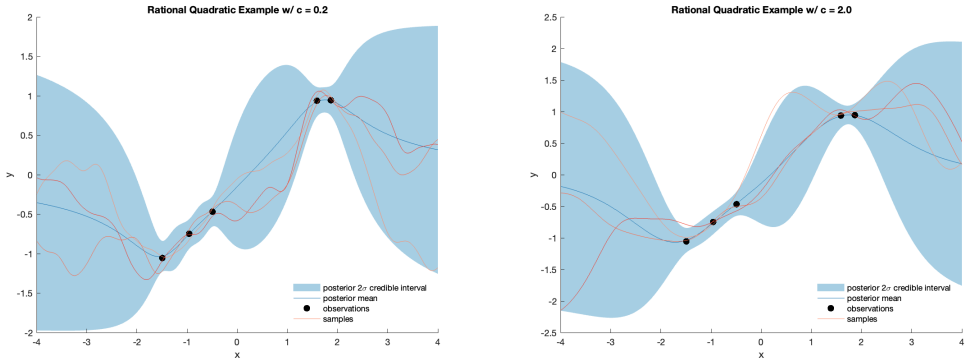
Another extension or variant of the squared exponential kernel is the rational quadratic kernel, which can be seen as a scale mixture of squared exponential kernels (i.e., a weighted sum of infinitely many squared exponentials), each with a different length scale. Intuitively, this kernel can be used to model smooth (i.e., infinitely differentiable) functions that vary across a variety of length scales i.e., functions that can change both very quickly or very slowly but always smoothly.

Formally, the rational quadratic kernel is given by

$$k(x, x') = \lambda^2 \left(1 + \frac{\|\mathbf{x} - \mathbf{x}'\|_2^2}{2c\ell}\right)^{-c}$$

where c is a hyperparameter that controls the relative preference for length scales closer to 0 vs. length scales closer to ℓ: smaller values of c result in a bias towards shorter length scales whereas larger values of c result in a bias towards longer length scales.

Figure 3 shows examples of a GP with a rational quadratic covariance function conditioned on the same dataset with two different values of c; note how the samples from the GP with $c = 2$ tend to vary over longer length scales than the samples from the GP with $c = 0.2$.



(a) $c = 0.2$ (b) $c = 2$

Figure 3: Two Gaussian processes fit to the same set of 5 observations using the rational quadratic kernel. The kernel hyperparameters are again set to be $\ell = \lambda = 1$ and 3 samples from each posterior are shown.

Automatic Relevance Determination

Suppose we have 2-dimensional inputs, $\mathbf{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$. In this setting, we can write out the norm in the squared exponential kernel as

$$k(\mathbf{x}, \mathbf{x}') = \lambda^2 \exp\left(-\frac{(x_1 - x_1')^2}{2\ell^2} - \frac{(x_2 - x_2')^2}{2\ell^2}\right)$$

Written out this way, one possible interpretation is that the dimensions x_1 and x_2 are constrained to have the same length scale, ℓ. While this may be appropriate in some settings, it may not always be the case that the modeled function behaves the same in both dimensions.

A natural extension in such settings is the squared exponential kernel with *automatic relevance determination*: this modification effectively gives each dimension its own length scale hyperparameter, allowing the GP belief to change at different rates along each dimension. Again, in two dimensions, this would look like:

$$k(\mathbf{x}, \mathbf{x}') = \lambda^2 \exp\left(-\frac{(x_1 - x_1')^2}{2\ell_1^2} - \frac{(x_2 - x_2')^2}{2\ell_2^2}\right)$$

Figure 4 shows an example of a 2-dimensional posterior GP mean with a squared exponential covariance function using automatic relevance determination; note how the mean changes rapidly as x_1 (the horizontal axis) changes and does not change very much as x_2 (the vertical axis) changes.

Roughly speaking, one can conceptualize the length scale as the amount of distance required between inputs in the corresponding distance in order for the belief about the function at those inputs to be uncorrelated. Therefore, one interpretation of a long length scale (relative to the input domain) is that the posterior mean is almost independent of the corresponding feature or that the feature is "irrelevant".

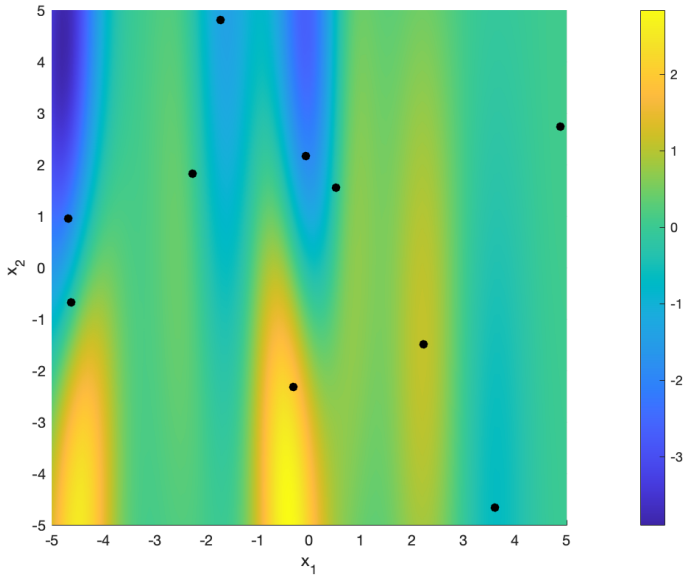


Figure 4: An example of a Gaussian process posterior mean using a squared exponential kernel with automatic relevance determination: the length scale in the x_1 dimension is $\ell_1 = 0.5$ while the length scale in the x_2 dimension is $\ell_2 = 5$. The black dots are the data points used to condition the posterior belief.

Thus, one possible use for kernels with automatic relevance determination is to determine which features are relevant for a given task (hence the name) or equivalently, which features are can be removed from the analysis. This can be done by first setting the kernel hyperparameters using something like maximum likelihood estimation and then identifying which length scales are relatively long and relatively short.

Covariance Functions for Discrete Data

Given discrete or categorical input features, one straightforward approach to building a kernel is to simply define the function for all possible pairs of input values e.g., if x can take on values $\{A, B, C\}$, then one way of describing a kernel over this variable is through a table:

$k(x, x') =$

$x \backslash x'$	A	B	C
A	k_{AA}	k_{AB}	k_{AC}
B	k_{AB}	k_{BB}	k_{BC}
C	k_{AC}	k_{BC}	k_{CC}

Unfortunately, this kernel has $O(n^2)$ hyperparameters which must be specified where n is the number of possible values x can take on. A simpler form for a kernel over a discrete variable is:

$$k(x, x') = \begin{cases} s^2 & \text{if } x = x' \\ cs^2 & \text{otherwise} \end{cases}$$

for $c < 1$ (to ensure that the resulting Gram matrices are positive semi-definite). Essentially, this kernel defines all non-identical pairs of values as equivalently dissimilar, significantly reducing the number of hyperparameters that need to be specified.

For multiple discrete features, the above protocol can be extended so that a kernel is defined over all possible pairs of combinations of features (for those of you so inclined, you can think of this as defining a kernel on the tensor product of the feature spaces). Of course, this approach will scale very poorly: if both x_1 and x_2 can take on one of three values a kernel K ($\mathbf{x} = [x_1, x_2], \mathbf{x}' = [x_1', x_2']$) defined this way will have $(9 + 1)9/2 = 45$ hyperparameters! This can be reduced using an approach like the one above, where the kernel treats all different values in each dimension x_i as equivalently dissimilar but that will also scale poorly as the number of dimensions (i.e., the number of discrete features) increases.

An alternative approach is to define a kernel for each discrete feature and then combine them (see below); the drawback here is that the resulting kernel will effectively ignore all interaction effects between the discrete features.

Combining Covariance Functions

What can be done if our data is mixed i.e., the input has both continuous and discrete features? Well it turns out that kernels can be combined in a few convenient ways:

Formally, if k_1 and k_2 are valid kernels then:

- $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}')$
- $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}')$

are also both valid kernels.

Returning to the scenario of mixed data, if the input vector x can be decomposed into $\mathbf{x} = [\mathbf{x}_c, \mathbf{x}_d]^T$ where $\mathbf{x}_c$ are all the continuous features and $\mathbf{x}_d$ are all the discrete features, then one possible kernel for this data is

$$k(\mathbf{x}, \mathbf{x}') = k_c(\mathbf{x}_c, \mathbf{x}_c') + k_d(\mathbf{x}_d, \mathbf{x}_d')$$

where k_c is some arbitrary kernel defined over continuous inputs and k_d is some arbitrary kernel defined over discrete inputs.

The ability to combine kernels can also be useful even when the domain is not mixed. Consider the dataset shown in Figure 5.

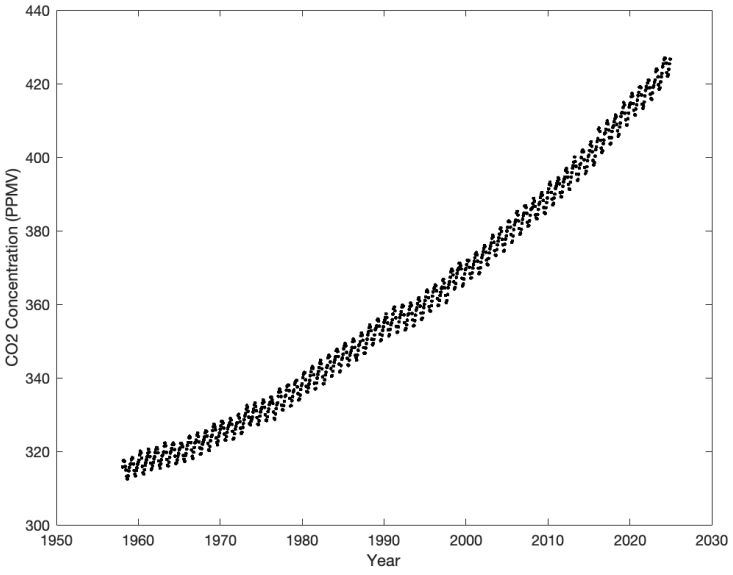


Figure 5: A sample dataset where the underlying function to be predicted displays multiple disparate trends/behaviors. This type of function is not particularly well modeled by any single covariance function and might benefit from a combination of different kernels.

This data set plots the monthly average atmospheric CO_2 concentrations in parts per million by volume (PPMV) at the Mauna Loa observatory in Hawaii (data found here: https://scrippsco2.ucsd.edu/data/atmospheric_co2/mlo.html).

Looking at the data, there are multiple trends:

- a long-term, smoothly increasing trend,
- a seasonal, periodic or repeating component,
- medium-term irregularities and,
- short-term irregularities or noise.

In GPML Section 5.4.3, Rasumussen and Williams suggest modeling each component with one or two kernel functions and then combining them into a single kernel. Specifically, they use the following kernels:

- a squared exponential kernel: $k_1(x, x') = \theta_1^2 \exp\left(-\frac{(x-x')^2}{2\theta_2^2}\right)$,
- a squared exponential kernel multiplied by a *periodic* kernel:
  $k_2(x, x') = \theta_3^2 \exp\left(-\frac{(x-x')^2}{2\theta_4^2} - \frac{2\sin(\pi(x-x'))}{\theta_5^2}\right)$,

- a rational quadratic kernel: $k_3(x, x') = \theta_6^2 \left(1 + \frac{(x-x')^2}{2\theta_8\theta_7^2}\right)^{\theta_8}$, and
- another squared exponential kernel: $k_4(x, x') = \theta_9^2 \exp\left(-\frac{(x-x')^2}{2\theta_{10}^2}\right)$.

The final kernel is simply $k(x, x') = k_1(x, x') + k_2(x, x') + k_3(x, x') + k_4(x, x')$; the hyperparameters $\{\theta_1, \theta_2, \ldots, \theta_{10}\}$ are fit to maximize the likelihood of the data between 1958 and 2003.

Figure 6 shows the posterior belief over the years 2003 to 2025 when fit on the data from 1958 to 2003. Note how the posterior has captured the seasonal trend but has underestimated the longer-term increases.

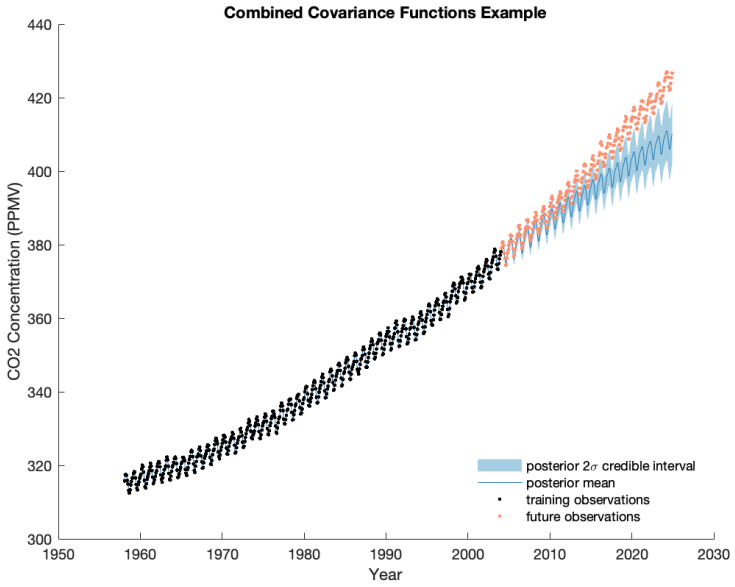


Figure 6: A sample dataset where the underlying function to be predicted displays multiple disparate trends/behaviors. This type of function is not particularly well modeled by any single covariance function and might benefit from a combination of different kernels.

Covariance Functions for Non-vector Data

So far, we have only considered GPs over vector data i.e., data that can be represented as $x = [x_1, ..., x_d]^T$. However, there are many interesting settings where the inputs are not vector inputs. GPs (and most other kernel methods for that matter) are convenient in these settings because they can be applied in a relatively straightforward way to non-vector inputs: all that's need is a kernel defined over the inputs i.e., some way of describing how similar or dissimilar any pair of inputs are. Two common and useful examples of non-vector inputs are strings and graphs.

String Kernels

A common class of string kernels involves looking at the frequency of substrings between two input strings; intuitively, the two input strings S and S' are similar (and therefore, should be more related under a GP model or have a higher covariance) if there are many substrings that frequently occur in both S and S'.

Let a string S be a collection of characters from some finite alphabet $\mathcal{A}$ and define $N_\sigma(S)$ as the number of times some substring σ occurs in the S; σ is a substring of S if $S = \rho\sigma\tau$ for some possibly empty strings ρ and τ where $\rho\sigma\tau$ denotes the concatenation of the strings ρ, σ and τ. Also let $|S|$ denote the length of the string S. Then the above kernel can be expressed as:

$$k(S, S') = \sum_{\sigma \in \mathcal{A}*} w_\sigma \left(N_\sigma(S) * N_\sigma(S')\right)$$

where the notation $\sigma \in \mathcal{A}*$ indicates that the sum is over all possible strings using characters in $\mathcal{A}$ (the $*$ is meant to indicate the regular expression quantifigre). While this is technically an infinite sum, observe that if $|\sigma| > max(|S|, |S'|)$, the term inside the sum will always be 0.

The hyperparameters for this kernel are the weights w_s; note that in this most general form, there are exponentially many hyperparameters that need to specified! Correspondingly, computing this kernel naïvely would require searching each input for exponentially many substrings. Fortunately, there are convenient algorithmic tricks for computing this kernel given certain settings of the weights (for those of you who are interested in such things, these algorithms make clever use of suffix trees to compute string kernels in $O(|S| + |S'|)$).

A few common ways of constraining these weights are

- The bag-of-characters kernel: $w_\sigma = 0$ if $|\sigma| > 1$
- The k-spectrum kernel: $w_\sigma = 0$ if $|\sigma| \neq k$
- The bag-of-words kernel: $w_s = 0$ if $\sigma \notin$ some corpus or dictionary
- The weights in a bag-of-words kernel can be set using some importance metric over words e.g., term frequency inverse document frequency or TF-IDF.

Graph Kernels

Graphs are a mathematical abstraction defined over a set of nodes or vertices where some pairs of vertices are connected by an edge. Graphs are used in a variety of machine learning tasks e.g., to represent physical/social networks or chemical structures.

Unfortunately, graphs are notoriously computationally complex and many straightforward approaches or definitions of similarity between graphs are known (or assumed) to be NP-complete e.g., the binary isomorphism kernel where $k(G, G')$ is 1 if the graphs are topographically identical and 0 otherwise.

A few common types of graph kernels are

- Walk-based kernels: given labels for the vertices in each graph, a kernel can be defined by performing a bunch of random walks in both graphs and comparing the number of walks (or sequences of vertices) that occurred in both graphs.
- Subgraph-based kernels: these kernels compare the counts of all types of subgraphs (usually of a small, fixed size e.g., all subgraphs of 3 or 4 vertices) in the input graphs.
- Neighborhood-based kernels: again, given vertex labels, a kernel can be defined by comparing how similar each vertex's neighborhood (i.e., the set of vertices adjacent to a given vertex) is across the two graphs.

Most graph kernels are very complex to formally define and compute so are beyond the scope of this course. However, if you are interested in learning more about this topic, I encourage you to look over the (optional) survey paper found in the readings/resources of this lecture.