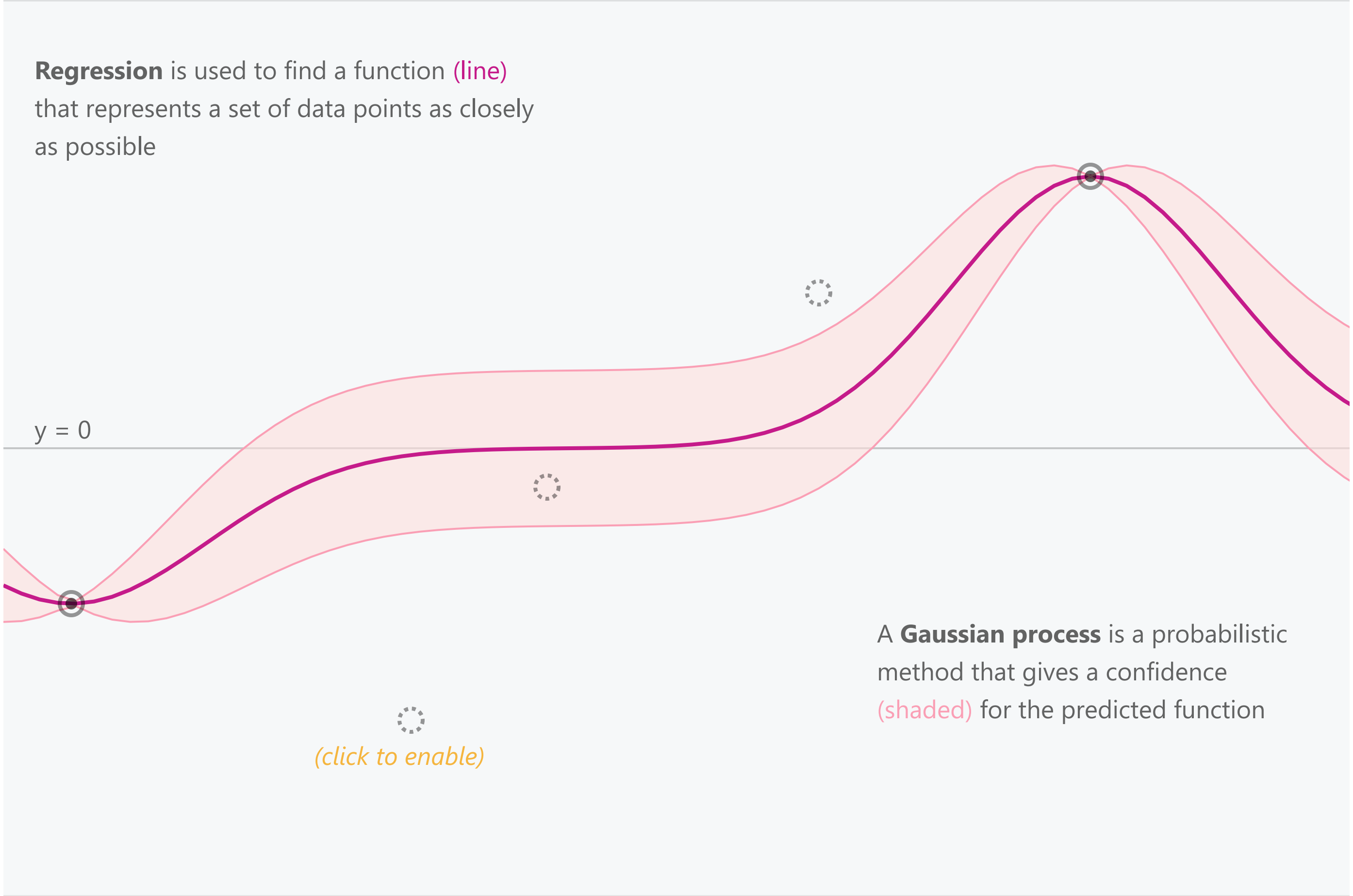
Distill ABOUT PRIZE SUBMIT

A Visual Exploration of Gaussian Processes

How to turn a collection of small building blocks into a versatile tool for solving regression problems.



Regression is used to find a function (line) that represents a set of data points as closely as possible

A **Gaussian process** is a probabilistic method that gives a confidence (shaded) for the predicted function

AUTHORS

Jochen Görtler
Rebecca Kehlbeck
Oliver Deussen

AFFILIATIONS

University of Konstanz
University of Konstanz
University of Konstanz

PUBLISHED

April 2, 2019

DOI

10.23915/distill.00017

Even if you have spent some time reading about machine learning, chances are that you have never heard of Gaussian processes. And if you have, rehearsing the basics is always a good way to refresh your memory. With this blog post we want to give an introduction to Gaussian processes and make the mathematical intuition behind them more approachable.

Gaussian processes are a powerful tool in the machine learning toolbox [1]. They allow us to make predictions about our data by incorporating prior knowledge. Their most obvious area of application is *fitting* a function to the data. This is called regression and is used, for example, in robotics or time series forecasting. But Gaussian processes are not limited to regression — they can also be extended to classification and clustering tasks [2] [3]. For a given set of training points, there are potentially infinitely many functions that fit the data. Gaussian processes offer an elegant solution to this problem by assigning a probability to each of these functions [1]. The mean of this probability distribution then represents the most probable characterization of the data. Furthermore, using a probabilistic approach allows us to incorporate the confidence of the prediction into the regression result.

We will first explore the mathematical foundation that Gaussian processes are built on — we invite you to follow along using the interactive figures and hands-on examples. They help to explain the impact of individual components, and show the flexibility of Gaussian processes. After following this article we hope that you will have a visual intuition on how Gaussian processes work and how you can configure them for different types of data.

Multivariate Gaussian distributions

Before we can explore Gaussian processes, we need to understand the mathematical concepts they are based on. As the name suggests, the Gaussian distribution (which is often also referred to as *normal* distribution) is the basic building block of Gaussian processes. In particular, we are interested in the multivariate case of this distribution, where each random variable is distributed normally and their joint distribution is also Gaussian. The multivariate Gaussian distribution is defined by a mean vector μ and a covariance matrix Σ. You can see an interactive example of such distributions in the figure below.

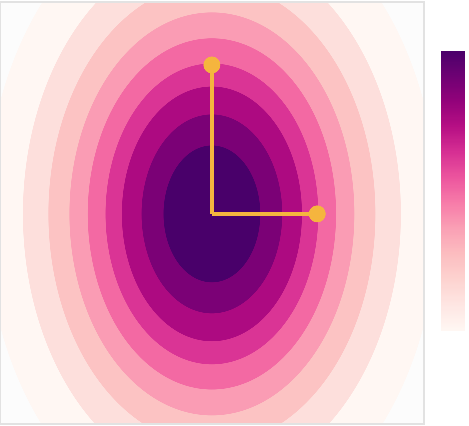
The mean vector μ describes the expected value of the distribution. Each of its components describes the mean of the corresponding dimension. Σ models the variance along each dimension and determines how the different random variables are correlated. The covariance matrix is always symmetric and positive semi-definite [4]. The diagonal of Σ consists of the variance σ_i^2 of the i-th random variable. And the off-diagonal elements σ_{ij} describe the correlation between the i-th and j-th random variable.

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} \sim \mathcal{N}(\mu, \Sigma)$$

We say X follows a normal distribution. The covariance matrix Σ describes the shape of the distribution. It is defined in terms of the expected value E:

$$\Sigma = \mathrm{Cov}(X_i, X_j) = E\left[(X_i - \mu_i)(X_j - \mu_j)^T\right]$$

Visually, the distribution is centered around the mean and the covariance matrix defines its shape. The following figure shows the influence of these parameters on a two-dimensional Gaussian distribution. The variances for each random variable are on the diagonal of the covariance matrix, while the other values show the covariance between them.



Covariance matrix (Σ)

1	0.7
0.7	2

By dragging the handles you can adjust the variance along each dimension, as well as the correlation between the two random variables. *Violet* values show a high probability inside the distribution.

Gaussian distributions are widely used to model the real world. For example, we can employ them to describe errors of measurements or phenomena under the assumptions of the *central limit theorem* [1]. In the next section we will take a closer look at how to manipulate Gaussian distributions and extract useful information from them.

Marginalization and Conditioning

Gaussian distributions have the nice algebraic property of being closed under conditioning and marginalization. Being closed under conditioning and marginalization means that the resulting distributions from these operations are also Gaussian, which makes many problems in statistics and machine learning tractable. In the following we will take a closer look at both of these operations, as they are the foundation for Gaussian processes.

Marginalization and *conditioning* both work on subsets of the original distribution and we will use the following notation:

$$P_{X,Y} = \begin{bmatrix} X \\ Y \end{bmatrix} \sim \mathcal{N}(\mu, \Sigma) = \mathcal{N}\left(\begin{bmatrix} \mu_X \\ \mu_Y \end{bmatrix}, \begin{bmatrix} \Sigma_{XX} \, \Sigma_{XY} \\ \Sigma_{YX} \, \Sigma_{YY} \end{bmatrix}\right)$$

With X and Y representing subsets of original random variables.

Through *marginalization* we can extract partial information from multivariate probability distributions. In particular, given a normal probability distribution $P(X,Y)$ over vectors of random variables X and Y, we can determine their marginalized probability distributions in the following way:

$$\begin{aligned} X &\sim \mathcal{N}(\mu_X, \Sigma_{XX}) \\ Y &\sim \mathcal{N}(\mu_Y, \Sigma_{YY}) \end{aligned}$$

The interpretation of this equation is that each partition X and Y only depends on its corresponding entries in μ and Σ. To marginalize out a random variable from a Gaussian distribution we can simply drop the variables from μ and Σ.

$$p_X(x) = \int_y p_{X,Y}(x,y)dy = \int_y p_{X|Y}(x|y)p_Y(y)dy$$

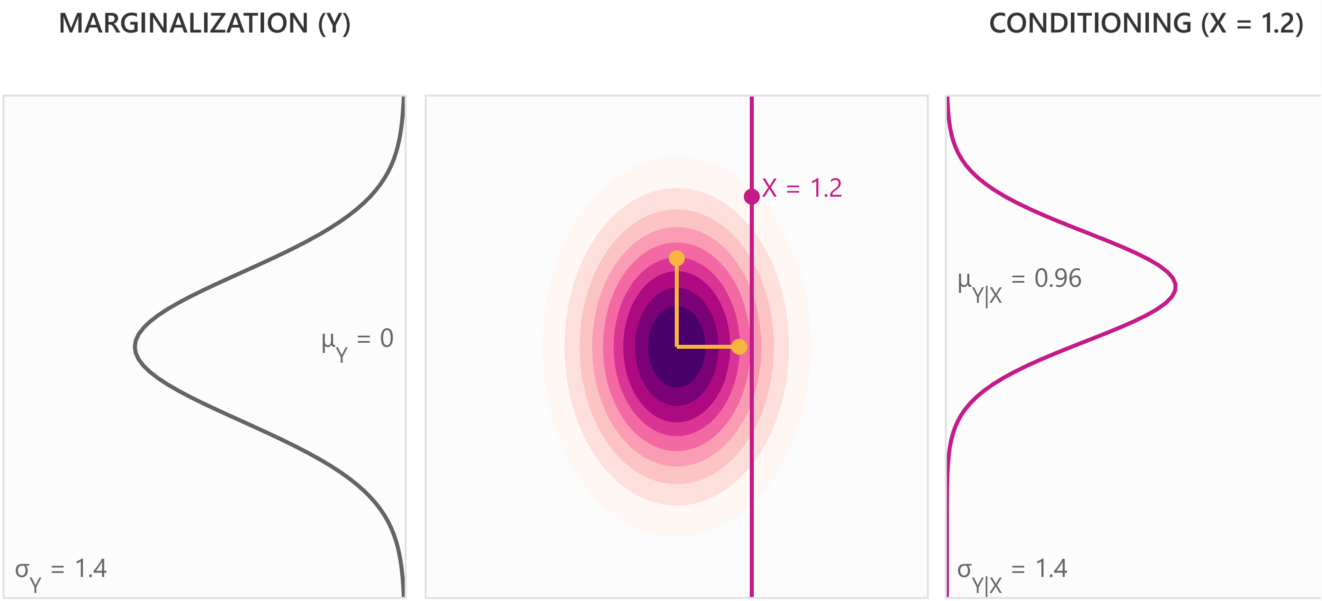
The way to interpret this equation is that if we are interested in the probability density of $X = x$, we need to consider all possible outcomes of Y that can jointly lead to the result [2] .

Another important operation for Gaussian processes is *conditioning*. It is used to determine the probability of one variable depending on another variable. Similar to marginalization, this operation is also closed and yields a modified Gaussian distribution. This operation is the cornerstone of Gaussian processes since it allows Bayesian inference, which we will talk about in the next section. Conditioning is defined by:

$$\begin{aligned} X|Y &\sim \mathcal{N}(\ \mu_X + \Sigma_{XY}\Sigma_{YY}^{-1}(Y - \mu_Y),\ \Sigma_{XX} - \Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX}\ ) \\ Y|X &\sim \mathcal{N}(\ \mu_Y + \Sigma_{YX}\Sigma_{XX}^{-1}(X - \mu_X),\ \Sigma_{YY} - \Sigma_{YX}\Sigma_{XX}^{-1}\Sigma_{XY}\ ) \end{aligned}$$

Note that the new mean only depends on the conditioned variable, while the covariance matrix is independent from this variable.

Now that we have worked through the necessary equations, we will think about how we can understand the two operations visually. While marginalization and conditioning can be applied to multivariate distributions of many dimensions, it makes sense to consider the two-dimensional case as shown in the following figure. Marginalization can be seen as integrating along one of the dimensions of the Gaussian distribution, which is in line with the general definition of the marginal distribution. Conditioning also has a nice geometric interpretation — we can imagine it as making a cut through the multivariate distribution, yielding a new Gaussian distribution with fewer dimensions.

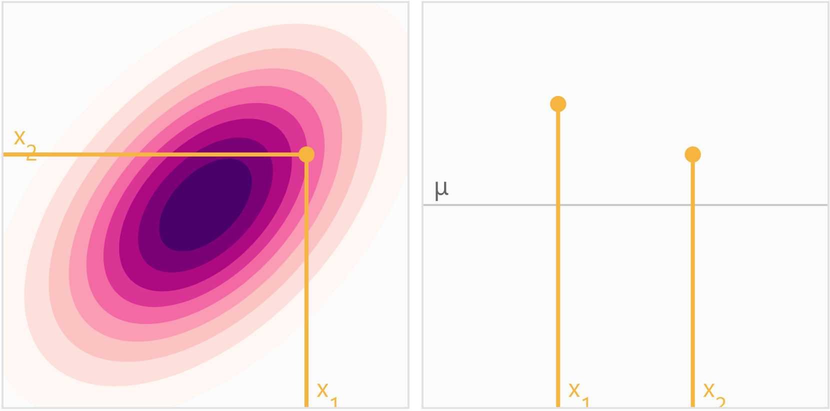


A bivariate normal distribution in the center. On the left you can see the result of marginalizing this distribution for Y, akin to integrating along the X axis. On the right you can see the distribution conditioned on a given X, which is similar to a cut through the original distribution. The Gaussian distribution and the conditioned variable can be changed by dragging the handles.

Gaussian Processes

Now that we have recalled some of the basic properties of multivariate Gaussian distributions, we will combine them together to define Gaussian processes, and show how they can be used to tackle regression problems.

First, we will move from the continuous view to the discrete representation of a function: rather than finding an implicit function, we are interested in predicting the function values at concrete points, which we call *test points* X. So how do we derive this functional view from the multivariate normal distributions that we have considered so far? Stochastic processes, such as Gaussian processes, are essentially a set of random variables. In addition, each of these random variables has a corresponding index i. We will use this index to refer to the i-th dimension of our n-dimensional multivariate distributions. The following figure shows an example of this for two dimensions:

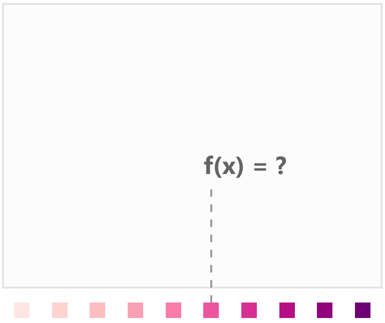


Here, we have a two-dimensional normal distribution. Each dimension x_i is assigned an index $i \in \{1, 2\}$. You can drag the handles to see how a particular sample (left) corresponds to functional values (right). This representation also allows us to understand the connection between the covariance and the resulting values: the underlying Gaussian distribution has a positive covariance between x_1 and x_2 — this means that x_2 will increases as x_1 gets larger and vice versa. You can also drag the handles in the figure to the right and observe the probability of such a configuration in the figure to the left.

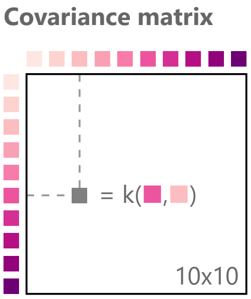
Now, the goal of Gaussian processes is to learn this underlying distribution from *training data*. Respective to the test data X, we will denote the training data as Y. As we have mentioned before, the key idea of Gaussian processes is to model the underlying distribution of X together with Y as a multivariate normal distribution. That means that the joint probability distribution $P_{X,Y}$ spans the space of possible function values for the function that we want to predict. Please note that this joint distribution of test and training data has $|X| + |Y|$ dimensions.

In order to perform regression on the training data, we will treat this problem as *Bayesian inference*. The essential idea of Bayesian inference is to update the current hypothesis as new information becomes available. In the case of Gaussian processes, this information is the training data. Thus, we are interested in the conditional probability $P_{X|Y}$. Finally, we recall that Gaussian distributions are closed under conditioning — so $P_{X|Y}$ is also distributed normally.

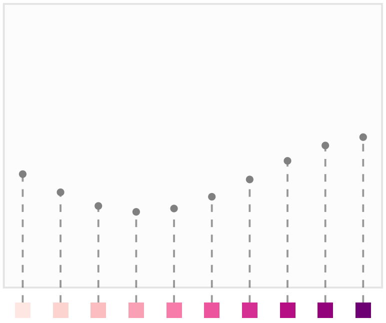
Now that we have the basic framework of Gaussian processes together, there is only one thing missing: how do we set up this distribution and define the mean μ and the covariance matrix Σ? The covariance matrix Σ is determined by its *covariance function* k, which is often also called the *kernel* of the Gaussian process. We will talk about this in detail in the next section. But before we come to this, let us reflect on how we can use multivariate Gaussian distributions to estimate function values. The following figure shows an example of this using ten test points at which we want to predict our function:



We are interested in predicting the function values for 10 different *x* values from [□,■] without knowing about training points.



The covariance matrix is created by pairwise evaluation of the kernel function resulting in a 10-dimensional distribution.



Sampling from this distribution results in a 10-dimensional vector where each entry represents one function value.

In Gaussian processes we treat each test point as a random variable. A multivariate Gaussian distribution has the same number of dimensions as the number of random variables. Since we want to predict the function values at $|X| = N$ test points, the corresponding multivariate Gaussian distribution is also N - dimensional. Making a prediction using a Gaussian process ultimately boils down to drawing samples from this distribution. We then interpret the i-th component of the resulting vector as the function value corresponding to the i-th test point.

Kernels

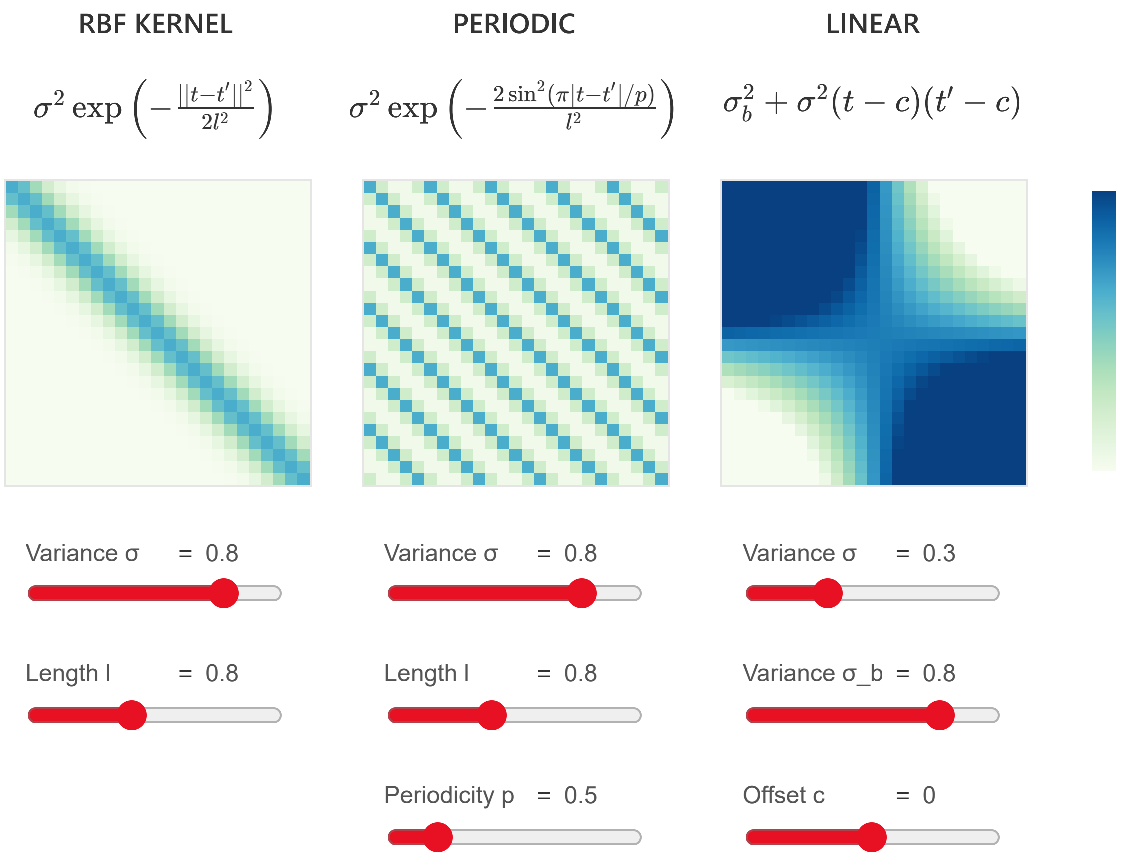
Recall that in order to set up our distribution, we need to define μ and Σ. In Gaussian processes it is often assumed that $\mu = 0$, which simplifies the necessary equations for conditioning. We can always assume such a distribution, even if $\mu \neq 0$, and add μ back to the resulting function values after the prediction step. This process is also called *centering* of the data. So configuring μ is straight forward — it gets more interesting when we look at the other parameter of the distribution.

The clever step of Gaussian processes is how we set up the covariance matrix Σ. The covariance matrix will not only describe the shape of our distribution, but ultimately determines the characteristics of the function that we want to predict. We generate the covariance matrix by evaluating the kernel k, which is often also called *covariance function*, pairwise on all the points. The kernel receives two points $t, t' \in \mathbb{R}^n$ as an input and returns a similarity measure between those points in the form of a scalar:

$$k : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}, \quad \Sigma = \text{Cov}(X, X') = k(t, t')$$

We evaluate this function for each pairwise combination of the test points to retrieve the covariance matrix. This step is also depicted in the figure above. In order to get a better intuition for the role of the kernel, let's think about what the entries in the covariance matrix describe. The entry Σ_{ij} describes how much influence the i-th and j-th point have on each other. This follows from the definition of the multivariate Gaussian distribution, which states that Σ_{ij} defines the correlation between the i-th and the j-th random variable. Since the kernel describes the similarity between the values of our function, it controls the possible shape that a fitted function can adopt. Note that when we choose a kernel, we need to make sure that the resulting matrix adheres to the properties of a covariance matrix.

Kernels are widely used in machine learning, for example in *support vector machines* [5]. The reason for this is that they allow similarity measures that go far beyond the standard euclidean distance ($L2$-distance). Many of these kernels conceptually embed the input points into a higher dimensional space in which they then measure the similarity [3]. The following figure shows examples of some common kernels for Gaussian processes. For each kernel, the covariance matrix has been created from $N = 25$ linearly-spaced values ranging from $[-5, 5]$. Each entry in the matrix shows the covariance between points in the range of $[0, 1]$.



RBF KERNEL

$\sigma^2 \exp\left(-\frac{||t-t'||^2}{2l^2}\right)$

Variance σ = 0.8

Length l = 0.8

PERIODIC

$\sigma^2 \exp\left(-\frac{2\sin^2(\pi|t-t'|/p)}{l^2}\right)$

Variance σ = 0.8

Length l = 0.8

Periodicity p = 0.5

LINEAR

$\sigma_b^2 + \sigma^2(t-c)(t'-c)$

Variance σ = 0.3

Variance σ_b = 0.8

Offset c = 0

This figure shows various kernels that can be used with Gaussian processes. Each kernel has different parameters, which can be changed by adjusting the according sliders. When grabbing a slider, information on how the current parameter influences the kernel will be shown on the right.

Kernels can be separated into *stationary* and *non-stationary* kernels. *Stationary* kernels, such as the RBF kernel or the periodic kernel, are functions invariant to translations, and the covariance of two points is only dependent on their relative position. *Non-stationary* kernels, such as the linear kernel, do not have this constraint and depend on an absolute location. The stationary nature of the RBF kernel can be observed in the banding around the diagonal of its covariance matrix (as shown in this figure). Increasing the length parameter increases the banding, as points further away from each other become more correlated. For the periodic kernel, we have an additional parameter P that determines the periodicity, which controls the distance between each repetition of the function. In contrast, the parameter C of the linear kernel allows us to change the point on which all functions hinge.

There are many more kernels that can describe different classes of functions, which can be used to model the desired shape of the function. A good overview of different kernels is given by Duvenaud [6]. It is also possible to combine several kernels — but we will get to this later.

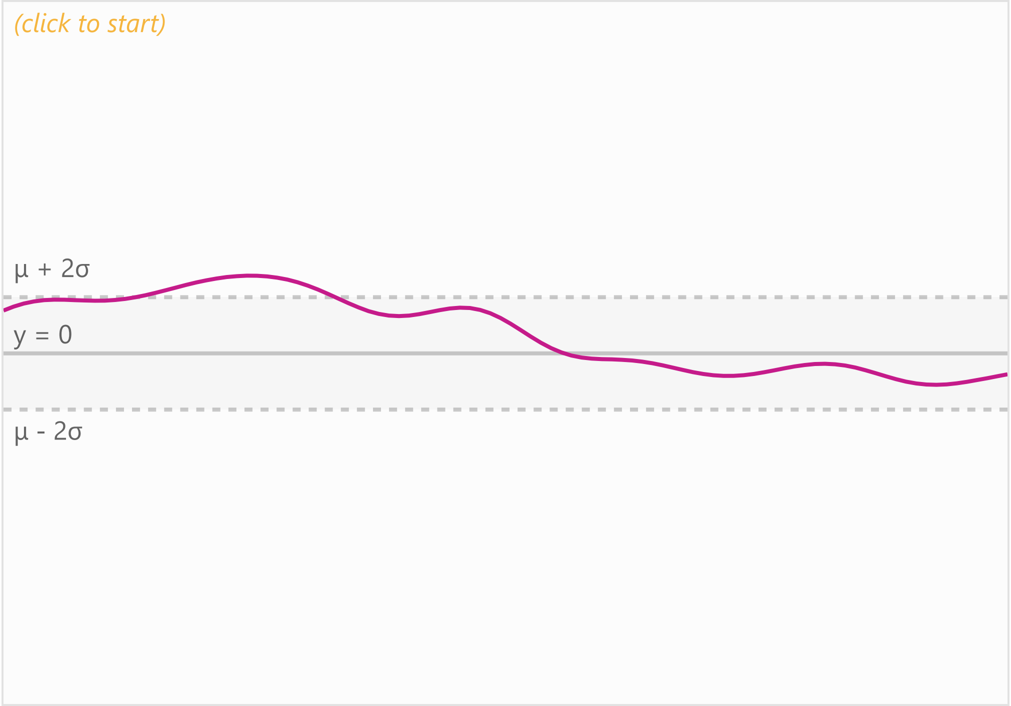
Prior Distribution

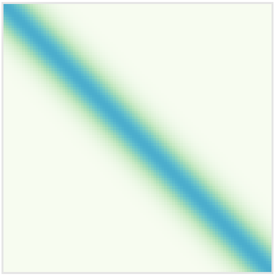
We will now shift our focus back to the original task of regression. As we have mentioned earlier, Gaussian processes define a probability distribution over possible functions. In this figure above, we show this connection: each sample of our multivariate normal distribution represents one realization of our function values. Because this distribution is a multivariate Gaussian distribution, the distribution of functions is normal. Recall that we usually assume $\mu = 0$. For now, let's consider the case where we have not yet observed any training data. In the context of Bayesian inference, this is called the *prior* distribution P_X.

If we have not yet observed any training examples, this distribution revolves around $\mu = 0$, according to our original assumption. The prior distribution will have the same dimensionality as the number of test points $N = |X|$. We will use the kernel to set up the covariance matrix, which has the dimensions $N \times N$.

In the previous section we have looked at examples of different kernels. The kernel is used to define the entries of the covariance matrix. Consequently, the covariance matrix determines which type of functions from the space of all possible functions are more probable. As the prior distribution does not yet contain any additional information, it is perfect to visualize the influence of the kernel on the distribution of functions. The following figure shows samples of potential functions from prior distributions that were created using different kernels:

● RBF ○ Periodic ○ Linear





Variance σ = 0.8

Length l = 0.8

Clicking on the graph results in continuous samples drawn from a Gaussian process using the selected kernel. After each draw, the previous sample fades into the background. Over time, it is possible to see that functions are distributed normally around the mean μ .

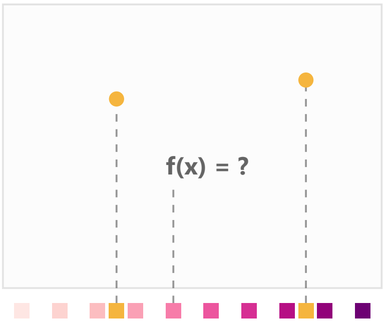
Adjusting the parameters allows you to control the shape of the resulting functions. This also varies the confidence of the prediction. When decreasing the variance σ, a common parameter for all kernels, sampled functions are more concentrated around the mean μ. For the *Linear* kernel, setting the variance $\sigma_b = 0$ results in a set of functions constrained to perfectly intersect the offset point c. If we set $\sigma_b = 0.2$ we can model uncertainty, resulting in functions that pass close to c.

Posterior Distribution

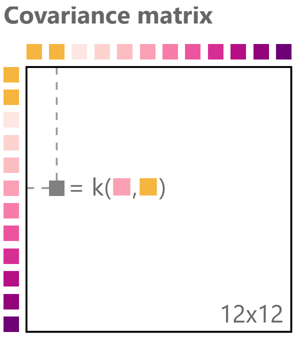
So what happens if we observe training data? Let's get back to the model of Bayesian inference, which states that we can incorporate this additional information into our model, yielding the *posterior* distribution $P_{X|Y}$. We will now take a closer look at how to do this for Gaussian processes.

First, we form the joint distribution $P_{X,Y}$ between the test points X and the training points Y. The result is a multivariate Gaussian distribution with dimensions $|Y| + |X|$. As you can see in the figure below, we concatenate the training and the test points to compute the corresponding covariance matrix.

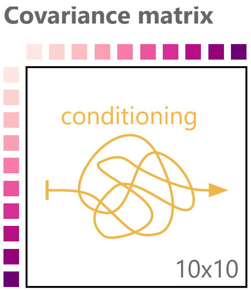
For the next step we need one operation on Gaussian distributions that we have defined earlier. Using *conditioning* we can find $P_{X|Y}$ from $P_{X,Y}$. The dimensions of this new distribution matches the number of test points N and the distribution is also normal. It is important to note that conditioning leads to derived versions of the mean and the standard deviation: $X|Y \sim \mathcal{N}(\mu', \Sigma')$. More details can be found in the related section on conditioning multivariate Gaussian distributions. The intuition behind this step is that the training points constrain the set of functions to those that pass through the training points.



Adding training points (🟧) changes the number of dimensions of the multivariate Guassian distribution.



The covariance matrix is created by pairwise evaluation of the kernel function resulting in a 12-dimensional distribution.



Through **conditioning** we obtain the distribution that describes the prediction of the function values for the given *x* values.

As mentioned before, the conditional distribution $P_{X|Y}$ forces the set of functions to precisely pass through each training point. In many cases this can lead to fitted functions that are unnecessarily complex. Also, up until now, we have considered the training points Y to be perfect measurements. But in real-world scenarios this is an unrealistic assumption, since most of our data is afflicted with measurement errors or uncertainty. Gaussian processes offer a simple solution to this problem by modeling the error of the measurements. For this, we need to add an error term $\epsilon \sim \mathcal{N}(0, \psi^2)$ to each of our training points:

$$Y = f(X) + \epsilon$$

We do this by slightly modifying the setup of the joint distribution $P_{X,Y}$:

$$P_{X,Y} = \begin{bmatrix} X \\ Y \end{bmatrix} \sim \mathcal{N}(0, \Sigma) = \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY} + \psi^2 I \end{bmatrix}\right)$$

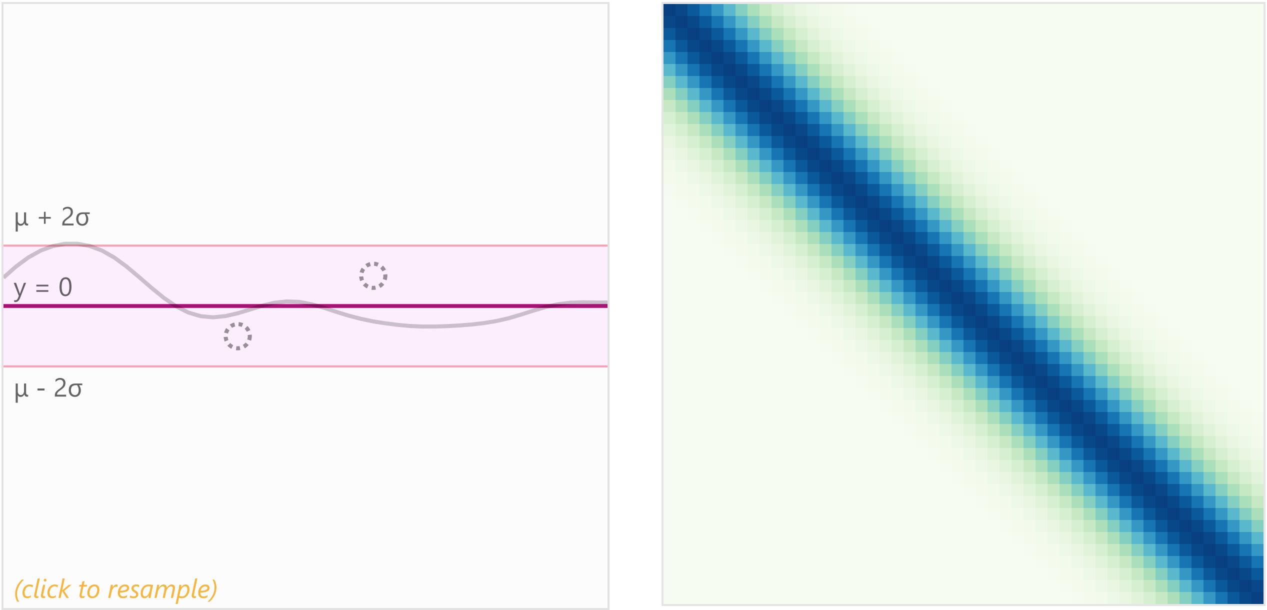
Again, we can use conditioning to derive the predictive distribution $P_{X|Y}$. In this formulation, ψ is an additional parameter of our model.

Analogous to the prior distribution, we could obtain a prediction for our function values by sampling from this distribution. But, since sampling involves randomness, the resulting fit to the data would not be deterministic and our prediction could end up being an outlier. In order to make a more meaningful prediction we can use the other basic operation of Gaussian distributions.

Through the *marginalization* of each random variable, we can extract the respective mean function value μ'_i and standard deviation $\sigma'_i = \Sigma'_{ii}$ for the i-th test point. In contrast to the prior distribution, where we set the mean to $\mu = 0$, the result of conditioning the joint distribution of test and training data will most likely have a non-zero mean $\mu' \neq 0$. Extracting μ' and σ' does not only lead to a more meaningful prediction, it also allows us to make a statement about the confidence of the prediction.

The following figure shows an example of the conditional distribution. At first, no training points have been observed. Accordingly, the mean prediction remains at 0 and the standard deviation is the same for each test point. By hovering over the covariance matrix you can see the influence of each point on the current test point. As long as no training points have been observed, the influence of neighboring points is limited locally.

The training points can be activated by clicking on them, which leads to a constrained distribution. This change is reflected in the entries of the covariance matrix, and leads to an adjustment of the mean and the standard deviation of the predicted function. As we would expect, the uncertainty of the prediction is small in regions close to the training data and grows as we move further away from those points.



Without any activated training data , this figure shows the prior distribution of a Gaussian process with a RBF kernel. When hovering over the covariance matrix, the opacity of the gradient shows the influence of a function value on its neighbours. The distribution changes when we observe training data. Individual points can be activated by clicking on them. The Gaussian process is then constrained to make functions, that intersect these data points, more probable. The best explanation of the training data is given by the updated mean function.

In the constrained covariance matrix, we can see that the correlation of neighbouring points is affected by the training data. If a predicted point lies on the training data, there is no correlation with other points. Therefore, the function must pass directly through it. Predicted values further away are also affected by the training data — proportional to their distance.

Combining different kernels

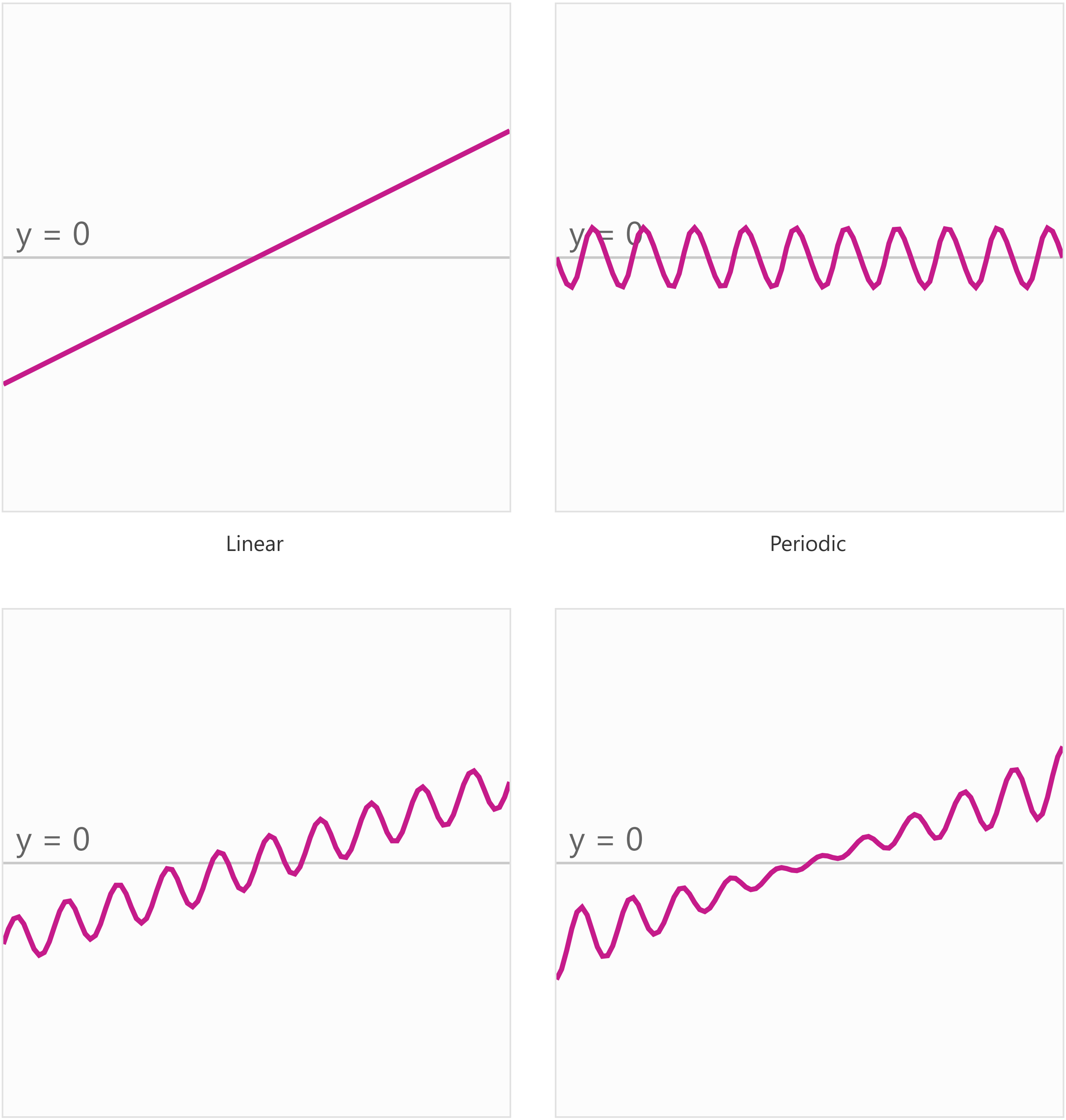
As described earlier, the power of Gaussian processes lies in the choice of the kernel function. This property allows experts to introduce domain knowledge into the process and lends Gaussian processes their flexibility to capture trends in the training data. For example, by choosing a suitable bandwidth for the RBF kernel, we can control how smooth the resulting function will be.

A big benefit that kernels provide is that they can be combined together, resulting in a more specialized kernel. The decision which kernel to use is highly dependent on prior knowledge about the data, e.g. if certain characteristics are expected. Examples for this would be stationary nature, or global trends and patterns. As introduced in the section on kernels, stationary means that a kernel is translation invariant and therefore not dependent on the index i. This also means that we cannot model global trends using a strictly stationary kernel. Remember that the covariance matrix of Gaussian processes has to be positive semi-definite. When choosing the optimal kernel combinations, all methods that preserve this property are allowed. The most common kernel combinations would be addition and multiplication.

Let’s consider two kernels, a linear kernel k_{lin} and a periodic kernel k_{per}, for example. This is how we would multiply the two:

$$k^*(t, t') = k_{\text{lin}}(t, t') \cdot k_{\text{per}}(t, t')$$

However, combinations are not limited to the above example, and there are more possibilities such as concatenation or composition with a function[7]. To show the impact of a kernel combination and how it might retain qualitative features of the individual kernels, take a look at the figure below. If we add a periodic and a linear kernel, the global trend of the linear kernel is incorporated into the combined kernel. The result is a periodic function that follows a linear trend. When combining the same kernels through multiplication instead, the result is a periodic function with a linearly growing amplitude away from linear kernel parameter c.



Linear

Periodic

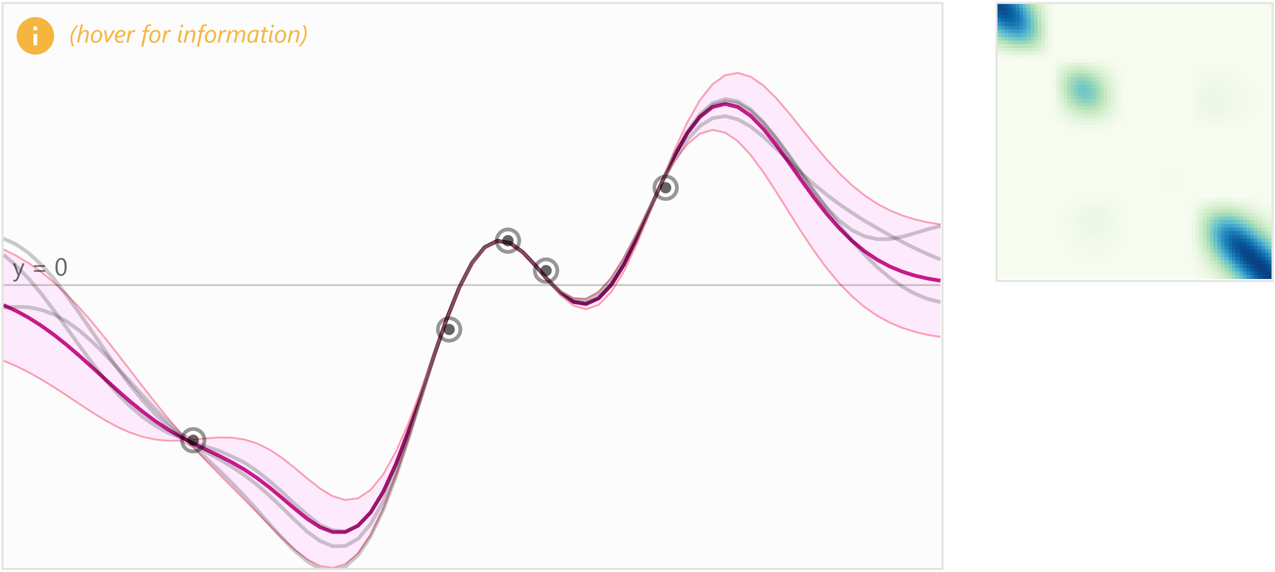
Linear + Periodic

Linear · Periodic

If we draw samples from a combined linear and periodic kernel, we can observe the different retained characteristics in the new sample. Addition results in a periodic function with a global trend, while the multiplication increases the periodic amplitude outwards.

Knowing more about how kernel combinations influence the shape of the resulting distribution, we can move on to a more complex example. In the figure below, the observed training data has an ascending trend with a periodic deviation. Using only a linear kernel, we can mimic a normal linear regression of the points. At first glance, the RBF kernel accurately approximates the points. But since the RBF kernel is stationary it will always return to $\mu = 0$ in regions further away from observed training data. This decreases the accuracy for predictions that reach further into the past or the future. An improved model can be created by combining the individual kernels through addition, which maintains both the periodic nature and the global ascending trend of the data. This procedure can be used, for example, in the analysis of weather data.

☑ RBF □ Periodic □ Linear



Using the checkboxes, different kernels can be combined to form a new Gaussian process. Only by using a combination of kernels, it is possible to capture the characteristics of more complex training data.

As discussed in the section about GPs, a Gaussian process can model uncertain observations. This can be seen when only selecting the linear kernel, as it allows us to perform linear regression even if more than two points have been observed, and not all functions have to pass directly through the observed training data.

Conclusion

With this article, you should have obtained an overview of Gaussian processes, and developed a deeper understanding on how they work. As we have seen, Gaussian processes offer a flexible framework for regression and several extensions exist that make them even more versatile.

For instance, sometimes it might not be possible to describe the kernel in simple terms. To overcome this challenge, learning specialized kernel functions from the underlying data, for example by using deep learning [8, 9], is an area of ongoing research. Furthermore, links between Bayesian inference, Gaussian processes and deep learning have been described in several papers [10, 11]. Even though we mostly talk about Gaussian processes in the context of regression, they can be adapted for different purposes, e.g. *model-peeling* and hypothesis testing. By comparing different kernels on the dataset, domain experts can introduce additional knowledge through appropriate combination and parameterization of the kernel.

If we have sparked your interest, we have compiled a list of further blog posts on the topic of Gaussian processes. In addition, we have linked two Python notebooks that will give you some hands-on experience and help you to get started right away.

Acknowledgments

We are very grateful to Carla Avolio and Marc Spicker for their feedback on the manuscript. In addition, we want to thank Jonas Körner for helping with the implementation of the figure explaining the multivariate Gaussian distribution. Furthermore, we would like to thank the German Research Foundation (DFG) for financial support within project A01 of the SFB-TRR 161 and within the Research Unit FOR 2111 with grant number DFG-431/16.

Discussion and Review

Review 1 - Anonymous
Review 2 - Anonymous
Review 3 - Austin Huang

Further Reading

The following blog posts offer more interactive visualizations and further reading material on the topic of Gaussian processes:

- Interactive visualization of Gaussian processes by ST John that joins together the different concepts introduced throughout this article.
- Gaussian process regression demo by Tomi Peltola
- Gaussian Processes for Dummies by Katherine Bailey
- Intuition behind Gaussian Processes by Mike McCourt
- Fitting Gaussian Process Models in Python by Chris Fonnesbeck
- A Practical Guide to Gaussian Processes by Marc Peter Deisenroth, Yicheng Luo, and Mark van der Wilk: heuristics for initializing and optimizing Gaussian processes

If you want more of a hands-on experience, there are also many Python notebooks available:

- Fitting Gaussian Process Models in Python by Chris Fonnesbeck
- Gaussian process lecture by Andreas Damianou

Footnotes

1. One of the implications of this theorem is that a collection of independent, identically distributed random variables with finite variance have a mean that is distributed normally. A good introduction to the central limit theorem is given by this video from Khan Academy. [↩]
2. The corresponding Wikipedia article has a good description of the marginal distribution, including several examples. [↩]
3. If the kernel follows Mercer's theorem it can be used to define a Hilbert space. More information on this can be found on Wikipedia. [↩]

References

1. **Gaussian Processes in Machine Learning** [PDF]
Rasmussen, C.E., 2004. Advanced Lectures on Machine Learning, pp. 63--71. Springer Berlin Heidelberg. DOI: 10.1007/978-3-540-28650-9_4
2. **Gaussian Processes for Object Categorization** [link]
Kapoor, A., Grauman, K., Urtasun, R. and Darrell, T., 2010. International Journal of Computer Vision, Vol 88(2), pp. 169--188. DOI: 10.1007/s11263-009-0268-3
3. **Clustering Based on Gaussian Processes** [PDF]
Kim, H. and Lee, J., 2007. Neural Computation, Vol 19(11), pp. 3088--3107. MIT Press - Journals. DOI: 10.1162/neco.2007.19.11.3088
4. **Covariance matrix (Encyclopedia of Mathematics)**
1994. Springer Science+Business Media B.V. / Kluwer Academic Publishers.
5. **The Nature of Statistical Learning Theory**
Vapnik, V.N., 2000. Springer New York. DOI: 10.1007/978-1-4757-3264-1
6. **Automatic model construction with Gaussian processes** [link]
Duvenaud, D., 2014. University of Cambridge. PhD Thesis.
7. **Information theory, inference and learning algorithms**
MacKay, D.J., 2003. Cambridge University Press.
8. **Using Deep Belief Nets to Learn Covariance Kernels for Gaussian Processes** [PDF]
Hinton, G.E. and Salakhutdinov, R.R., 2008. Advances in Neural Information Processing Systems 20, pp. 1249--1256. Curran Associates, Inc.
9. **Deep Kernel Learning** [PDF]
Wilson, A.G., Hu, Z., Salakhutdinov, R. and Xing, E.P., 2016. Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, Vol 51, pp. 370--378.
10. **Deep Neural Networks as Gaussian Processes** [PDF]
Lee, J., Sohl-Dickstein, J., Pennington, J., Novak, R., Schoenholz, S. and Bahri, Y., 2018. International Conference on Learning Representations.
11. **Deep Gaussian Processes** [PDF]
Damianou, A. and Lawrence, N., 2013. Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, Vol 31, pp. 207--215. PMLR.

Updates and Corrections

If you see mistakes or want to suggest changes, please create an issue on GitHub.

Reuse

Citation

For attribution in academic contexts, please cite this work as

```
Görtler, et al., "A Visual Exploration of Gaussian Processes", Distill, 2019.
```

BibTeX citation

```
@article{görtler2019a,
  author = {Görtler, Jochen and Kehlbeck, Rebecca and Deussen, Oliver},
  title = {A Visual Exploration of Gaussian Processes},
  journal = {Distill},
  year = {2019},
  note = {https://distill.pub/2019/visual-exploration-gaussian-processes},
  doi = {10.23915/distill.00017}
}
```

Distill is dedicated to clear explanations of machine learning

About Submit Prize Archive RSS GitHub Twitter ISSN 2476-0757