CSE517A Machine Learning **Spring 2022**

Lecture 10: Gaussian Processes

Instructor: Quan Nguyen

Reading: FCML 8-8.2.4, 8.2.7 (GP Regression); GPML 2.1 (Weight-space View), 2.2 (Function-space View)

1 Multivariate Gaussian distributions

We first review the definition and properties of Gaussian distribution:

A Gaussian random variable $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, where $\boldsymbol{\mu}$ is the mean and Σ is the co-variance matrix, has the following probability density function:

$$p(\mathbf{x}; \boldsymbol{\mu}, \Sigma) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} e^{-\frac{1}{2}((\mathbf{x}-\boldsymbol{\mu})^\top \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu}))}$$

where $|\Sigma|$ is the determinant of Σ.

The Gaussian distribution occurs very often in real world data. This is probably because of the *Central Limit Theorem* (CLT). The CLT states that, given some conditions, the arithmetic mean of $m > 0$ samples is approximately normal distributed - independent of the sample distribution.

(1) Normalization:

$$\int_{\mathbf{x}} p(\mathbf{x}; \boldsymbol{\mu}, \Sigma) \mathrm{d}\mathbf{x} = 1$$

(2) Summation:

If $\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ and $\mathbf{y}' \sim \mathcal{N}(\boldsymbol{\mu}', \Sigma')$, then

$$\mathbf{y} + \mathbf{y}' \sim \mathcal{N}(\boldsymbol{\mu} + \boldsymbol{\mu}', \Sigma + \Sigma')$$

Now, let $\mathbf{y}$ be Gaussian random vector $\mathbf{y} = \begin{bmatrix} \mathbf{y}_A \\ \mathbf{y}_B \end{bmatrix}$, with mean $\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_A \\ \boldsymbol{\mu}_B \end{bmatrix}$ and co-variance matrix $\Sigma = \begin{bmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{bmatrix}$. We have the following properties:

(3) Marginalization:

The *marginal distributions* $p(\mathbf{y}_A) = \int_{\mathbf{y}_B} p(\mathbf{y}_A, \mathbf{y}_B; \boldsymbol{\mu}, \Sigma) \mathrm{d}\mathbf{y}_B$ and $p(\mathbf{y}_B) = \int_{\mathbf{y}_A} p(\mathbf{y}_A, \mathbf{y}_B; \boldsymbol{\mu}, \Sigma) \mathrm{d}\mathbf{y}_A$ are Gaussian:

$$\mathbf{y}_A \sim \mathcal{N}(\boldsymbol{\mu}_A, \Sigma_{AA})$$
$$\mathbf{y}_B \sim \mathcal{N}(\boldsymbol{\mu}_B, \Sigma_{BB})$$

(4) Conditioning:

The *conditional distribution* of $\mathbf{y}_A$ on $\mathbf{y}_B$

$$p(\mathbf{y}_A \mid \mathbf{y}_B) = \frac{p(\mathbf{y}_A, \mathbf{y}_B; \boldsymbol{\mu}, \Sigma)}{\int_{\mathbf{y}_A} p(\mathbf{y}_A, \mathbf{y}_B; \boldsymbol{\mu}, \Sigma) \mathrm{d}\mathbf{y}_A}$$

is also Gaussian:

$$\mathbf{y}_A \mid \mathbf{y}_B \sim \mathcal{N}\left(\mu_A + \Sigma_{AB}\Sigma_{BB}^{-1}(\mathbf{y}_B - \boldsymbol{\mu}_B), \Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}\right)$$

This property will be useful in deriving Gaussian Process predictions.

2 Introduction

Recap: Linear regression model

$$y = f(\mathbf{x}) + \varepsilon = \mathbf{w}^\top \mathbf{x} + \varepsilon$$

where we assume zero-mean i.i.d. Gaussian noise:

$$\varepsilon \sim \mathcal{N}(0, \sigma_n^2).$$

So, the *predictive distribution given the model parameters* is Gaussian:

$$p(y \mid \mathbf{x}, \mathbf{w}) = \mathcal{N}(\mathbf{w}^\top \mathbf{x}, \sigma_n^2)$$

We can make two extensions for the linear regression model:

(1) Regularization (Ridge Regression)

We model $\mathbf{w}$ as a random variable and assume a *prior distribution over parameters* $p(\mathbf{w})$. Then, the *posterior over parameters* is given by:

$$p(\mathbf{w} \mid X, \mathbf{y}) = \frac{p(\mathbf{y} \mid X, \mathbf{w})p(\mathbf{w})}{p(\mathbf{y} \mid X)} = \frac{\textit{likelihood} \times \textit{prior}}{\textit{marginal likelihood}}$$

For ridge regression specifically, we assume a *Gaussian prior* over parameters $\mathbf{w}$:

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \sigma_p^2 I)$$

(2) Feature Space Transformation

$$y = f(\mathbf{x}) + \varepsilon = \mathbf{w}^\top \phi(\mathbf{x}) + \varepsilon$$

Using the *kernel trick* leads us to kernel ridge regression:

$$y = f(\mathbf{x}) + \varepsilon = \sum_{i=1}^{n} \alpha_i \mathrm{k}(\mathbf{x}_i, \mathbf{x}) + \varepsilon$$

Notation: We will call $\mathcal{X}$ *input space* and $\phi(\mathcal{X})$ *feature space*. Hence, *feature space* refers to the **new transformed** feature space explicitly defined by ϕ or implicitly through $k(.,.)$.

Putting (1) and (2) Together

Instead of thinking about regularization and feature space transformations separately, we can derive a model directly by looking at the *predictive distribution*:

$$p(y^* \mid \mathbf{x}^*, X, \mathbf{y}) = \int_{\mathbf{w}} p(y^* \mid \mathbf{x}^*, \mathbf{w})\, p(\mathbf{w} \mid X, \mathbf{y}\, \mathrm{d}\mathbf{w}$$

This is essentially a (typically infinite) sum over the *predictive distributions* (cf. Eq. (2) but in feature space) using all possible model parameters $\mathbf{w}$ weighted by the probability of the respective parameter (*posterior*

[1] http://www.gaussianprocess.org/gpml/chapters/RW2.pdf

over parameters; cf. Eq. (2), but again in feature space).

Unexpectedly, Eq. (2) is actually tractable if $p(y^* \mid \mathbf{x}^*, \mathbf{w})$ and $p(\mathbf{w} \mid X, \mathbf{y})$ are both Gaussian, which can be achieved by using our *Gaussian noise* model in Eq. (2), and a *Gaussian likelihood* and a *Gaussian prior* in the feature space equivalent of Eq. (2). In that case, we have (without derivation[1]):

$$p(y^* \mid \mathbf{x}^*, \underbrace{X, \mathbf{y}}_{=D}) = \mathcal{N}\left(\boldsymbol{\mu}_{y^*|D}, \Sigma_{y^*|D}\right)$$

where

$$\boldsymbol{\mu}_{y^*|D} = K_*^\top (K + \sigma_n^2 I)^{-1}\mathbf{y} \tag{1}$$
$$\Sigma_{y^*|D} = K_{**} - K_*^\top (K + \sigma_n^2 I)^{-1} K_* \tag{2}$$

with

$$\begin{aligned} K &= \mathrm{k}(X, X) \\ K_* &= \mathrm{k}(X, \mathbf{x}^*) \\ K_{**} &= \mathrm{k}(\mathbf{x}^*, \mathbf{x}^*) \end{aligned}$$

X is the training set and $\mathbf{x}^*$ is the input test point. If there are multiple test inputs X_*, then $K_{**} = \mathrm{k}(X_*, X_*)$. This is a **Gaussian Process regression model** we can use for machine learning. It extends the kernel ridge regression model with an entire predictive distribution giving us a principled way to model *predictive uncertainty*!

Homework: verify that using Eq.(1) for predictions is exactly kernel ridge regression as derived in Lecture 9. HINT: use the fact that the *mean* of a Gaussian distribution is also it's *mode*.

A different (maybe *nicer*) way to derive Gaussian process regression is to think about modeling f directly (instead of y)! This is allso referred to as the *function-space view*.

3 Gaussian Processes

Problem: f is a (infinite-dimensional) function, but multivariate Gaussians are finite-dimensional.

Solution: Let's extend multivariate Gaussians to infinite dimensions!

Definition 3.1. Gaussian Process (GP):

A GP is a (potentially infinite) collection of random variables (RVs) such that the joint distribution of *every* finite subset of RVs is multivariate Gaussian:

$$f \sim \mathcal{GP}(\mu, \mathrm{k})$$

where $\mu(\mathbf{x})$ is the *mean function* and $\mathrm{k}(\mathbf{x}, \mathbf{x}')$ is the *co-variance function*.

4 Gaussian Process Regression (GPR)

Now, in order to use GPs for machine learning we need a prediction model. So, let's model the predictive distribution $p(f_* \mid \mathbf{x}_*, D)$ following the Bayesian approach using a GP prior:

$$f \mid \mathbf{x} \sim \mathcal{GP}(\mu(\mathbf{x}), \mathrm{k}(\mathbf{x}, \mathbf{x}))$$

and condition it on the training data D.

[1]For a derivation of Eqs. (2-2), we refer to GPML 2.1.1-2.1.2 (*weight-space view*).

4.1 Noise-free Observations

For simplicity let's start with noise free observations: $D = \{X, \mathbf{f}\}$. Note, we use $\mathbf{f}$ to denote the vector of noise-free training observations.

For D, we have

$$p(\mathbf{f} \mid X) = \mathcal{N}(\mu(X), \mathrm{k}(X, X)).$$

For $(\mathbf{x}_*, f_*)$, where f_* denotes the noise-free test target variable, we have

$$p(f_* \mid \mathbf{x}_*) = \mathcal{N}(\mu(\mathbf{x}_*), \mathrm{k}(\mathbf{x}_*, \mathbf{x}_*)).$$

Now, let's model the *joint distribution* of f_* and $\mathbf{f}$:

$$p\left(\begin{bmatrix} \mathbf{f} \mid X \\ f_* \mid \mathbf{x}_* \end{bmatrix}, \right) = \mathcal{N}\left(\begin{bmatrix} \mu(X) \\ \mu(\mathbf{x}_*) \end{bmatrix}, \begin{bmatrix} \mathrm{k}(X, X) & \mathrm{k}(X, \mathbf{x}_*) \\ \mathrm{k}(\mathbf{x}_*, X) & \mathrm{k}(\mathbf{x}_*, \mathbf{x}_*) \end{bmatrix}\right)$$

Conditioning on $\mathbf{f} \mid X$ gives us the GP posterior:

$$p(f_* \mid \mathbf{x}_*, X, \mathbf{f}) = \mathcal{N}\left(\mu(\mathbf{x}_*) + \mathrm{k}(\mathbf{x}_*, X)\mathrm{k}(X, X)^{-1}(\mathbf{f} - \mu(X)), \mathrm{k}(\mathbf{x}_*, \mathbf{x}_*) - \mathrm{k}(\mathbf{x}_*, X)\mathrm{k}(X, X)^{-1}\mathrm{k}(X, \mathbf{x}_*)\right)$$

Now, we can use the mean for predictions (*predictive mean*):

$$\begin{aligned} \bar{f}_* &= \mu(\mathbf{x}_*) + \mathrm{k}(\mathbf{x}_*, X)\mathrm{k}(X, X)^{-1}(\mathbf{f} - \mu(X)) \\ &= \mu(\mathbf{x}_*) + \underbrace{K_*^\top K^{-1}(\mathbf{f} - \mu(X))}_{=\sum_{i=1}^n \alpha_i \mathrm{k}(\mathbf{x}_*, \mathbf{x}_i)} \end{aligned} \tag{3}$$

where $\alpha_i = K^{-1}(\mathbf{f} - \mu(X))$. The *predictive variance*

$$cov(f_*) = K_{**} - K_*^\top K^{-1} K_* \tag{4}$$

gives us an idea of how (un)certain our predictions are.

4.2 Noisy Observations

For noisy observations we use $D = \{X, \mathbf{y}\}$, where $\mathbf{y} = f(X) + \varepsilon = \mathbf{f} + \varepsilon$. If we assume the noise to be **independent** and **zero-mean Gaussian** $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$, we have $p(\mathbf{y} \mid \mathbf{f}) = \mathcal{N}(\mathbf{f}, \sigma_n^2 I)$.[2] Now,

$$p(\mathbf{y} \mid X) = p(\mathbf{f} + \varepsilon \mid X) = \mathcal{N}(\mu_{\mathbf{f}|X} + \mu_\varepsilon, \Sigma_{\mathbf{f}|X} + \Sigma_\varepsilon) = \mathcal{N}(\mu(X), \mathrm{k}(X, X) + \sigma_n^2 I)$$

and we get:

$$p\left(\begin{bmatrix} \mathbf{y} \mid X \\ f_* \mid \mathbf{x}_* \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \mu(X) \\ \mu(\mathbf{x}_*) \end{bmatrix}, \begin{bmatrix} \mathrm{k}(X, X) + \sigma_n^2 I & \mathrm{k}(X, \mathbf{x}_*) \\ \mathrm{k}(\mathbf{x}_*, X) & \mathrm{k}(\mathbf{x}_*, \mathbf{x}_*) \end{bmatrix}\right).$$

In the same way as before, conditioning on $\mathbf{y} \mid X$ leads to the following *predictive mean* and *predictive variance* for the GP posterior $p(f_* \mid \mathbf{x}_*, X, \mathbf{y})$:

$$\bar{f}_* = \mu(\mathbf{x}_*) + \mathrm{k}(\mathbf{x}_*, X)(\mathrm{k}(X, X) + \sigma_n^2 I)^{-1}(\mathbf{y} - \mu(X)) \tag{5}$$

$$cov(f_*) = \mathrm{k}(\mathbf{x}_*, \mathbf{x}_*) - \mathrm{k}(\mathbf{x}_*, X)(\mathrm{k}(X, X) + \sigma_n^2 I)^{-1} k(X, \mathbf{x}_*) \tag{6}$$

So, for noisy observations, we essentially added $\sigma_n^2 I$ to $\mathrm{k}(X, X)$.

[2] If we model the dependency of y and f differently (as for instance for classification problems) or assume non-Gaussian noise, the GP posterior is no longer Gaussian and there is most likely no closed form solution for predictions. Approximate inference techniques can be used to overcome this. Covered in CSE515T or FCML 8.3 and 8.5.

4.3 Noisy Observations ans Noisy Predictions

For noisy predictions modeled by $p(y_* \mid \mathbf{x}_*, X, \mathbf{y})$ we have to add another σ_n^2 to $cov(f_*)$. Derivation omitted. Again, for predictions we can use the *predictive mean* $\bar{f}_*$ and additionally we get the *predictive variance* as a measure of confidence/(un)certainty about the point prediction. This predictive variance is now larger than for noise-free predictions. Note that the noisy prediction is the same as the noise-free prediction. Both should make sense intuitively.

For all three cases, if we have multiple test cases, i.e., X_* is a matrix, we also get the co-variance between their predictions Σ_*.

Note: *Magic* of GPs

Even though we model the Gaussian process as a *distribution over functions* (infinite-dimensional vectors), we only ever evaluate it on finite dimensional subsets of training and test points. So, effectively we only ever deal with multi-variate Gaussian distributions.

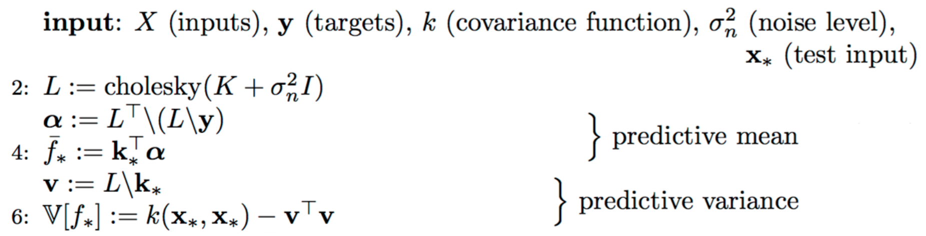
5 GPR in Practice

5.1 Practical Implementation

For a practical implementation of Gaussian process regression (GPR), we need a couple of ingredients:

- let's use a zero-mean GP $\rightarrow$ re-scale y_i's to have zero mean
- use the Cholesky decomposition of $K + \sigma_n^2 I = LL^\top$ to compute the inverse more efficiently (L is a *lower triangular matrix* and can be computed in $\mathcal{O}(\frac{n^3}{6})$)

Algorithm:



```
   input: $X$ (inputs), $\mathbf{y}$ (targets), $k$ (covariance function), $\sigma_n^2$ (noise level),
                                                              $\mathbf{x}_*$ (test input)
2: $L := \text{cholesky}(K + \sigma_n^2 I)$
   $\boldsymbol{\alpha} := L^\top \backslash (L \backslash \mathbf{y})$          } predictive mean
4: $\bar{f}_* := \mathbf{k}_*^\top \boldsymbol{\alpha}$
   $\mathbf{v} := L \backslash \mathbf{k}_*$                          } predictive variance
6: $\mathbb{V}[f_*] := k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^\top \mathbf{v}$
```

Figure 1: From GMPL Algorithm 2.1

The equation in line 3 are two systems of linear equations with triangular matrices, which can both be solved in $\mathcal{O}(\frac{n^2}{2})$ using *forward substitution* resp. *backward substitution*.

5.2 Hyperparameter Learning and Marginal Likelihood

Even though GPR is a non-parametric model, the covariance function K has parameters. Those parameters are so-called *hyperparameters* and they need to be learned from the training data. To see this consider the RBF kernel (cf. Lectures 8-9) with different values for its *length-scale hyperparameter*:

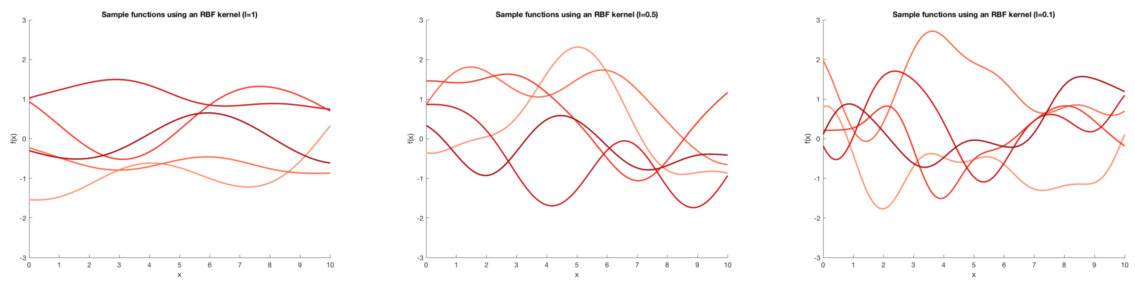


Figure 2: Illustration of GP priors using an RBF kernel with length-scale parameter ℓ (cf. *kernel width* parameter) decreasing from left to right.

Let $\boldsymbol{\theta}$ be the vector of all parameters of K, then $K = K_\theta$ and we can learn θ by **maximizing the marginal likelihood** $P(\mathbf{y} \mid X, \boldsymbol{\theta})$, cf. equations (2.28)-(2.30) in GPML Chapter 2 (p. 19).

For standard GPR the marginal likelihood (actually we use the *log marginal likelihood*) can be derived analytically. It turns out that for a Gaussian prior and Gaussian likelihood, the marginal likelihood is also Gaussian with mean $\boldsymbol{\mu}$ and covariance $K + \sigma^2 I$. Now, use Eq. (1) for its derivation (*derive as homework*). Then, we can compute its derivatives (*derive as homework*) and use any *out-of-the-box* solver such as (stochastic) gradient descent, or Rasmussen's minimize used in the GMPL MATLAB toolbox (caution: minimize the *negative* log marginal likelihood). Note that the marginal likelihood is *not convex* in its parameters and hence you will most likely get a **local minima/maxima**. To make this procedure more robust, you can rerun your optimization algorithm multiple times with different initialization and pick the lowest/highest return value.

5.3 Covariance Functions (Kernels) - The Heart of the GP Model

GPs gain a lot of their predictive power by selecting the *right* covaraince/kernel function. Selecting the covaraince function is the model selection process in the GP learning phase. There are three different ways to come up with a good covariance function (cf. GPML Chapter 5[3]):

- Expert knowledge (awesome to have – difficult to get)
- Bayesian model selection (more possibly analytically intractable integrals!!)
- Cross-validation (time consuming – but simple to implement)

Kernels used in Practice

Next to the ones already discussed in our lectures on kernel methods, there are many other covariance functions/kernels we can use in practice, see GPML 4.2 and https://www.cs.toronto.edu/~duvenaud/cookbook/ for an overview.

A popular and powerful covariance function to keep in mind is the **squared exponential** with **different length scales** for *each* input dimension:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T M (\mathbf{x} - \mathbf{x}')\right) + \sigma_n^2 \delta_{\mathbf{x},\mathbf{x}'},$$

with $M = diag(\boldsymbol{\ell})^{-2}$, where σ_f^2 is the signal variance, $\boldsymbol{\ell}$ is a vector of length-scale parameters (one for each input dimension), and σ_n^2 is the noise variance. This kernel is also called ARD kernel (for *automatic relevance determination*).

[3] http://www.gaussianprocess.org/gpml/chapters/RW5.pdf

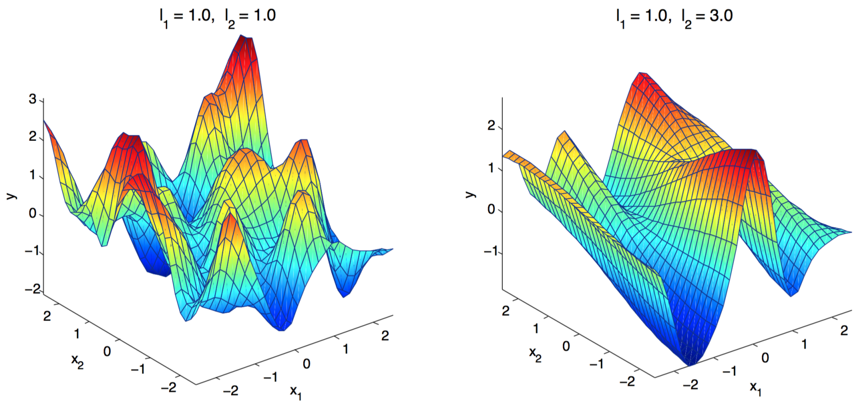


Figure 3: Illustration of GP priors using an ARD kernel with length-scale parameters ℓ_1 and ℓ_2.

Another trick is to build expressive **composite covariance functions** (mostly sums and products), such as in the Mauna Loa regression example (GPML 5.4.3) or what we did in our work on traffic frequency prediction.[4]

6 Summary

- GPs are an elegant and powerful ML method - they are truly *Bayesian*
- we get a measure of (un)certainty for the predictions *for free*
- GPs work very well for regression problems with small to medium sized training data
- we need to learn the kernel hyperparmeters and choose a good kernel/covariance function using the training data
- test points having more similar features with training points get more certain predictions (see Figure 4)

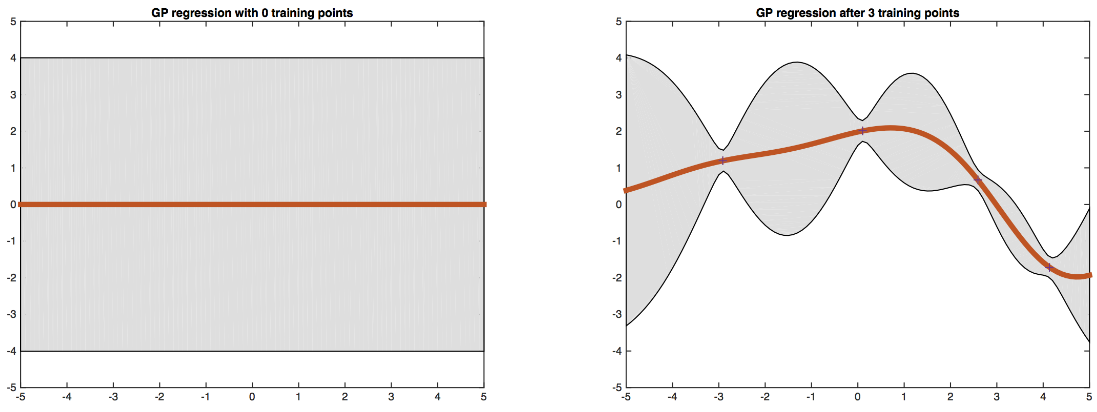


Figure 4: Example of GP regression: GP prior (left) and GP posterior (right). The mean function (*predictive mean*) is shown in red and the uncertainty (*predictive variance*) is shown as shaded gray area (two standard deviations).

[4]https://www.researchgate.net/publication/220765646_Stacked_Gaussian_Process_Learning

Extensions:

- run time $O(n^3) \to$ matrix inversion (gets slow when n is large) $\Rightarrow$ use sparse GPs for large n
- GPs are a little harder to realize for classification (we use the logistic likelihood which is non-Gaussian)
- we can model non-Gaussian likelihoods for regression, e.g., for count data (Poisson likelihood)
- non-Gaussian likelihoods or priors require approximate inference techniques