CSE517A Machine Learning **Spring 2025**

Lecture 8: Gaussian Processes

Instructor: Marion Neumann

Reading: FCML 8-8.2.4, 8.2.7 (GP Regression); GPML 2.1 (Weight-space View), 2.2 (Function-space View)[1]

1 Introduction

Recap: Linear regression model with zero-mean i.i.d. Gaussian noise:

$$y = f(\mathbf{x}) + \varepsilon = \mathbf{w}^\top \mathbf{x} + \varepsilon, \text{ with } \varepsilon \sim \mathcal{N}(0, \sigma_n^2) \tag{1}$$

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

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

And the *likelihood* is also (multivariate) Gaussian:

$$p(\mathbf{y} \mid X, \mathbf{w}) = \mathcal{N}(X^\top \mathbf{w}, \sigma_n^2 I) \tag{3}$$

Now, let's make three extensions to this model: **regularization** and **feature space transformation**.

(1) Regularization (MAP Estimation)

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:

$$p(\mathbf{w} \mid X, \mathbf{y}) = \frac{p(\mathbf{y} \mid X, \mathbf{w})\, p(\mathbf{w})}{p(\mathbf{y} \mid X)} = \frac{\textit{likelihood} \times \textit{prior}}{\textit{marginal likelihood}} \tag{4}$$

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

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

Exercise 1.1. Verify that the posterior over parameters $p(\mathbf{w} \mid X, \mathbf{y})$, Eq. (4), is indeed Gaussian when using the linear regression model with Gaussian noise and a Gaussian prior over parameters.

(2) Predictive Distribution

$$p(y^* \mid \mathbf{x}^*, X, \mathbf{y}) = \int_{\mathbf{w}} p(y^* \mid \mathbf{x}^*, \mathbf{w})\, p(\mathbf{w} \mid X, \mathbf{y})\, \mathrm{d}\mathbf{w} \tag{5}$$

This is essentially a (typically infinite) sum over the *predictive distributions given the parameters* (cf. Eq. (2)).

(3) 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:

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

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

Putting (1), (2), and (3) Together

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

$$p(y^* \mid \mathbf{x}^*, X, \mathbf{y}) = \int_{\tilde{\mathbf{w}}} p(y^* \mid \mathbf{x}^*, \tilde{\mathbf{w}})\, p(\tilde{\mathbf{w}} \mid X, \mathbf{y})\, \mathrm{d}\tilde{\mathbf{w}} \tag{6}$$

This is Eq. (5), where we view $\tilde{\mathbf{w}}$ in feature space using all possible (possibly infinite-dimensional) **model parameters $\tilde{\mathbf{w}}$ in feature space** weighted by the probability of the respective parameter (*posterior over parameters* as in Eq. (4), but now in feature space).

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

A Gaussian Process Regression Model

Unexpectedly, Eq. (6) is actually tractable if $p(y^* \mid \mathbf{x}^*, \tilde{\mathbf{w}})$ and $p(\tilde{\mathbf{w}} \mid X, \mathbf{y})$ are both Gaussian, which can be achieved by using our *Gaussian noise* model in Eq. (1), and a *Gaussian likelihood* and a *Gaussian prior* in the feature space equivalent of Eq. (4). In that case, the predictive distribution is also *Gaussian*.

→ **Derivation omitted**[2] ←

Predictive distribution:

$$p(y^* \mid \mathbf{x}^*, \underbrace{X, \mathbf{y}}_{=D}) = \mathcal{N}\left(\boldsymbol{\mu}_{y^*|D}, \Sigma_{y^*|D}\right) \tag{7}$$

where

$$\boldsymbol{\mu}_{y^*|D} = K_*^\top (K + \sigma_n^2 I)^{-1} \mathbf{y} \tag{8}$$

$$\Sigma_{y^*|D} = K_{**} - K_*^\top (K + \sigma_n^2 I)^{-1} K_* \tag{9}$$

with

$$K = \mathrm{k}(X, X) \in \mathbb{R}^{n \times n}$$
$$K_* = \mathrm{k}(X, \mathbf{x}^*) \in \mathbb{R}^n$$
$$K_{**} = \mathrm{k}(\mathbf{x}^*, \mathbf{x}^*) \in \mathbb{R}$$

X are the training inputs, $\mathbf{x}^*$ is the input test point, and $\mathrm{k}(\cdot,\cdot)$ is the *kernel function*. If there are multiple test inputs X_*, then $K_{**} = \mathrm{k}(X_*, X_*)$ and $K_* = \mathrm{k}(X, X_*)$ are matrices. 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*!

[2] For a derivation of Eqs. (7-9), we refer to GPML 2.1.1-2.1.2 (*weight-space view*).

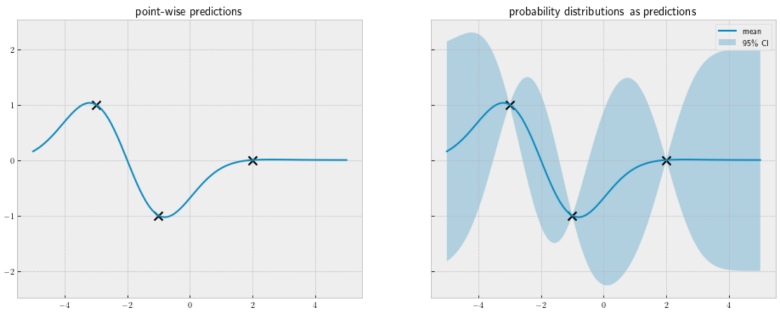

Figure 1: Illustration of predictions using point estimates (left) versus using predictive distributions (right), i.e., its mean (cf. Eq. (8)) and covariance (cf. Eq. (9)) leading to the **confidence ranges** shown as the 95% credible intervals.

Exercise 1.2. Verify that using Eq.(8) used for **point predictions** is exactly kernel ridge regression as derived in Lecture 7. HINT: use the fact that the *mean* of a Gaussian distribution is also it's *mode*.

2 Gaussian Processes

Let's start over. Forget about the previous section, since we didn't really derive the GPR model. A different (and in my opinion *nicer*) way to (actually) derive Gaussian process regression is to think about modeling the function $f(\cdot)$ directly (instead of the targets y)! This is also referred to as the *function-space view*.

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

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

Definition 2.1. ***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:

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

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

3 Gaussian Process Regression (GPR)

Now, in order to perform regression using GPs in a supervised machine learning setting we need a prediction model.

Model assumptions: Let's model the predictive distribution $p(f_* \mid \mathbf{x}_*, D)$ following the Bayesian approach using a GP *prior* specified by its mean and covariance/kernel function:

$$f(\mathbf{x}) = f \mid \mathbf{x} \sim \mathcal{GP}(\mu(\mathbf{x}), \mathrm{k}(\mathbf{x}, \mathbf{x})) \tag{10}$$

and then **condition it on the training data** D to get the GP *posterior*.

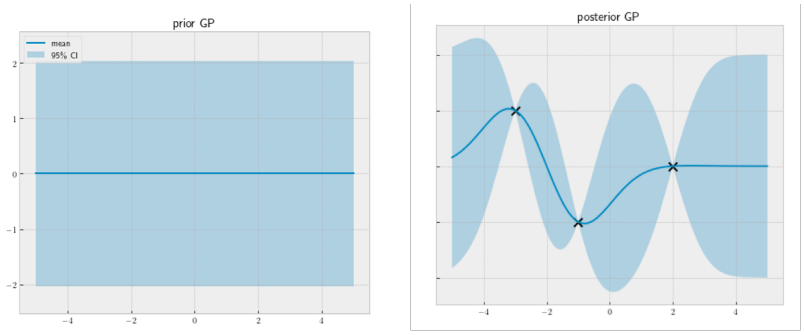

Figure 2: Example of GP regression using a zero-mean function and an RBF kernel: GP prior (left) and GP posterior (right). The mean function (*predictive mean*) is shown in dark-blue and the uncertainty (*predictive variance*) is shown as shaded light-blue area (two standard deviations – 95% credible interval).

Insight: With our new Gaussian process framework we can now **model the full joint distribution** of all training and test observations (f_i and f_* in the noise-free case and y_i and f_* or y_i and y_* in the case where we observe noisy targets) given the $\mathbf{x}$'s rather than having to assume that the y_i's are independent given the input feature $\mathbf{x}_i$ and the model parameters $\mathbf{w}$ (as we did in Lecture 4: MLE and MAP – cf. assumption (2)).

Note that with the GP framework we can model both noise-free observations f and noisy observations y:

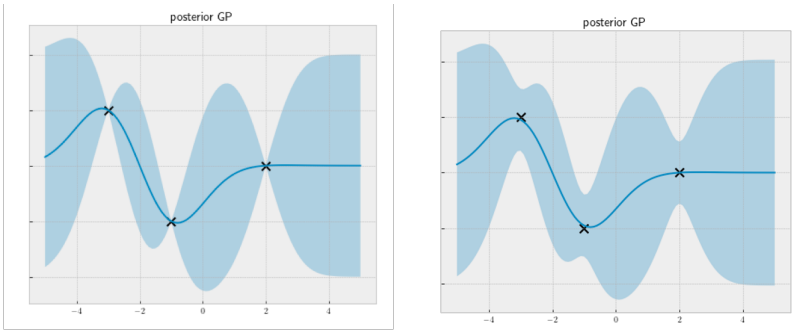

Figure 3: Example of GP regression using a zero-mean function and an RBF kernel. Left: GP posterior for noise-free observations (f). Right: GP posterior for noisy observations (y). The mean function (*predictive mean*) is shown in dark blue and the uncertainty (*predictive variance*) is shown as shaded light-blue area (two standard deviations).

Let's derive GPR for noise-free and then noisy observations:

3.1 Noise-free Observations

For simplicity let's start with noise free observations f, so our training data is $D = \{X, \mathbf{f}\}$. Note, we use $\mathbf{f} \in \mathbb{R}^n$ to denote the vector of noise-free training observations.

Applying our GP prior (Eq. (10)) to the training data D we get:

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

Applying our GP prior to test data point $(\mathbf{x}_*, f_*)$, where f_* denotes the noise-free test target variable, we get the *prior predictive distribution*:

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

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 $\mathbf{f} \mid X$ (see Appendix Eq. (A.7)) gives us the GP *posterior*:

$$f_D(\mathbf{x}) = f \mid \mathbf{x}, X, \mathbf{f} \sim \mathcal{GP}\left(\mu(\mathbf{x}) + \mathrm{k}(\mathbf{x},X)\mathrm{k}(X,X)^{-1}(\mathbf{f}-\mu(X)), \mathrm{k}(\mathbf{x},\mathbf{x}) - \mathrm{k}(\mathbf{x},X)\mathrm{k}(X,X)^{-1}\mathrm{k}(X,\mathbf{x})\right) \quad (11)$$

And in turn the *posterior predictive distribution*:

$$p(f_* \mid \mathbf{x}_*, X, \mathbf{f}) = \mathcal{N}\left(\mu(\mathbf{x}_*) + \mathrm{k}(\mathbf{x}_*,X)\mathrm{k}(X,X)^{-1}(\mathbf{f}-\mu(X)), \mathrm{k}(\mathbf{x}_*,\mathbf{x}_*) - \mathrm{k}(\mathbf{x}_*,X)\mathrm{k}(X,X)^{-1}\mathrm{k}(X,\mathbf{x}_*)\right) \quad (12)$$

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

$$\begin{aligned}\bar{f}_* = E[f_*] &= \mu(\mathbf{x}_*) + \mathrm{k}(\mathbf{x}_*,X)\mathrm{k}(X,X)^{-1}(\mathbf{f}-\mu(X))\\ &= \mu(\mathbf{x}_*) + K_*^\top K^{-1}(\mathbf{f}-\mu(X))\\ &= \mu(\mathbf{x}_*) + \sum_{i=1}^{n}\alpha_i \mathrm{k}(\mathbf{x}_*,\mathbf{x}_i) \quad \text{where } \alpha_i = K^{-1}(\mathbf{f}-\mu(X)).\end{aligned} \quad (13)$$

And the *predictive variance* gives us an idea of how **(un)certain** our prediction for f_* is:

$$cov(f_*) = V[f_*] = K_{**} - K_*^\top K^{-1} K_*. \quad (14)$$

Exercise 3.1. Assume we place a Gaussian process belief over a function $f(x)$ (in other words we model $f(x)$ as a GP) and would like to predict the function value y^* at test point $\mathbf{x}^*$. The prior distribution of this label y^* is a *normal distribution* with predictive mean μ^*_{prior} and predictive variance $(\sigma^*_{\text{prior}})^2$.

(a) Write down this prior distribution $p(y^* \mid \mathbf{x}^*)$.

(b) Assume we are allowed to observe the function value $y_1 = f(\mathbf{x}_1)$ at a single training point $\mathbf{x}_1$. This gives us the following posterior distribution of the test label y^*, which is still a *normal distribution* with an updated predictive mean $\mu^*_{\text{post}_1}$ and an updated predictive variance $(\sigma^*_{\text{post}_1})^2$:

$$p\big(y^* \mid \mathbf{x}^*, (\mathbf{x}_1, y_1)\big) = \mathcal{N}\big(\mu^*_{\text{post}_1}, (\sigma^*_{\text{post}_1})^2\big).$$

Given that μ_1 and σ_1^2 are the prior mean and prior variance of the label y_1, respectively, work out the updated predictive mean $\mu^*_{\text{post}_1}$ and updated predictive variance $(\sigma^*_{\text{post}_1})^2$. Hint: Remember that you only have **one** training point, so your solution will involve only scalars (and not matrices).

(c) Prove that if the observed label y_1 is exactly equal to the prior mean μ_1, the posterior predictive mean at test point $\mathbf{x}^*$ does not change from the prior predictive mean.

(d) Prove that the posterior predictive variance at test point $\mathbf{x}^*$ does not exceed the prior predictive variance. That is, our uncertainty about the test label y^* does not increase after observing a data point.

3.2 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 $p(\mathbf{y} \mid \mathbf{f}) = \mathcal{N}(\mathbf{f}, \sigma_n^2 I)$.[3] Now, using Eq. (A.3) we get:

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

[3] 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.

and hence:

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

In the same way as before (using Eq. (A.7)), conditioning on $\mathbf{y} \mid X$ leads to the following *predictive mean* and *predictive variance* for the GP posterior $p(f_* \mid \mathbf{x}_*, X, \mathbf{y})$:

$$\bar{f}_* = \mu(\mathbf{x}_*) + \mathrm{k}(\mathbf{x}_*, X)(\mathrm{k}(X,X) + \sigma_n^2 I)^{-1}(\mathbf{y} - \mu(X)) \tag{15}$$

$$cov(f_*) = \mathrm{k}(\mathbf{x}_*, \mathbf{x}_*) - \mathrm{k}(\mathbf{x}_*, X)(\mathrm{k}(X,X) + \sigma_n^2 I)^{-1} k(X, \mathbf{x}_*) \tag{16}$$

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

Exercise 3.2. Verify that Eq.(15) with a zero-mean function ($\mu(\mathbf{x}) = 0$) used for point predictions is equivalent to Eq.(8) and hence equivalent to kernel ridge regression as derived in Lecture 7.

3.3 Noisy Observations and Noisy Predictions

To get noisy predictions y_* (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.

3.4 Multiple Test Cases

For all three cases, if we have multiple test cases, say m test cases $\mathbf{f}_*$ or $\mathbf{y}_* \in \mathbb{R}^m$, i.e., X_* is a matrix, we also get the full co-variance between the test predictions $\Sigma_* \in \mathbb{R}^{m \times m}$.

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 finite-dimensional multi-variate Gaussian distributions**.

4 GPR in Practice

4.1 Practical Implementation

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

(1) Re-scale your y_i's to have a **mean of zero** to use a zero-mean GP (i.e., $\mu(\mathbf{x}) = 0$). This will simplify the prediction equation used: Eq. (13) or (15).

(2) Use the Cholesky decomposition of $K + \sigma_n^2 I = LL^\intercal$ to compute the inverse more efficiently (L is a *lower triangular matrix* and can be computed in $\mathcal{O}(\frac{n^3}{6})$).

(3) Instead of computing the inverse matrix directly, solve systems of linear equations (this can be done very efficiently in $\mathcal{O}(\frac{n^2}{2})$ for triangular matrices using *forward substitution* for lower triangular matrices and *backward substitution* for upper triangular matrices respectively).

Algorithm:

input: X (inputs), $\mathbf{y}$ (targets), k (covariance function), σ_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 4: From GMPL Algorithm 2.1

The equation in line 3 are two systems of linear equations ($A\mathbf{x} = \mathbf{b} \Leftrightarrow \mathbf{x} = A \backslash \mathbf{b}$) with triangular matrices L and $L^\top$ respectively.

4.2 Training a GP

Even though GPR is a non-parametric model, we still have to learn what specific covariance function to use. To turn Gaussian processes into powerful practical tool for supervised prediction it is essential to address the model selection problem. Model selection includes both, the discrete choice of the **functional form for the covariance function/kernel** as well as finding the optimal values for any **kernel hyperparameters**.

Cross-validation

In general, we can use **cross-validation** estimating the generalization performance for both kernel selection and hyperparameter tuning.

Hyperparameter Learning and Marginal Likelihood

It truns out the GPs (under specific conditions) come with a *sleek optimization process* to find the optimal *hyperparameters*. Let's remind ourselves how those hyperparaemters look like for an example: the **full covariance function** using the RBF kernel and input-independent noise:

$$\mathrm{k}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \, e^{-\frac{1}{2\ell^2}\|\mathbf{x}-\mathbf{x}'\|^2} + \sigma_n^2 \delta_{\mathbf{x},\mathbf{x}'}.$$

Note that in total we have three hyperparameters:

- the *length-scale parameter* ℓ
- the *output-scale parameter* σ_f^2 that gets multiplied to the kernel function
- the *noise parameter* σ_n^2 that gets added to the kernel function

Note that σ_f^2 also acts as a weight when using a sum of multiple kernels. Figure 5 illustrates the effect of different values for the *length-scale hyperparameter* ℓ.

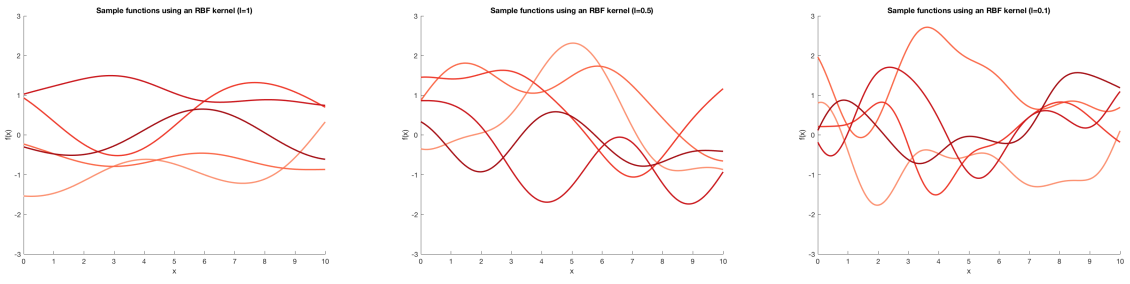

Figure 5: Illustration of GP priors using an RBF kernel with length-scale parameter ℓ (cf. *kernel width* parameter) decreasing from left to right.

Let $\boldsymbol{\theta}$ be the vector of all parameters of K. To indicate that k(.,.) evaluated on the training data depends on θ let's write $K = K_\theta$. Now, we can learn θ by **maximizing the marginal likelihood** $p(\mathbf{y} \mid X, \boldsymbol{\theta})$:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} p(\mathbf{y} \mid X, \boldsymbol{\theta}).$$

It turns out that for the standard GPR setting with Gaussian prior and Gaussian likelihood, the **marginal likelihood** (actually we use the *log marginal likelihood*) is also Gaussian with mean $\boldsymbol{\mu}$ and covariance $K = K_\theta$. This means that the marginal likelihood can be derived analytically (use Eq. (A.1) for its derivation). In addition, we get closed-form partial derivatives ☺ [→ cf. Homework 3].

To get the hyperparameters, the derivatives can be plugged into any *out-of-the-box* gradient based optimizer 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.

4.3 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[4]):

- 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*).

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

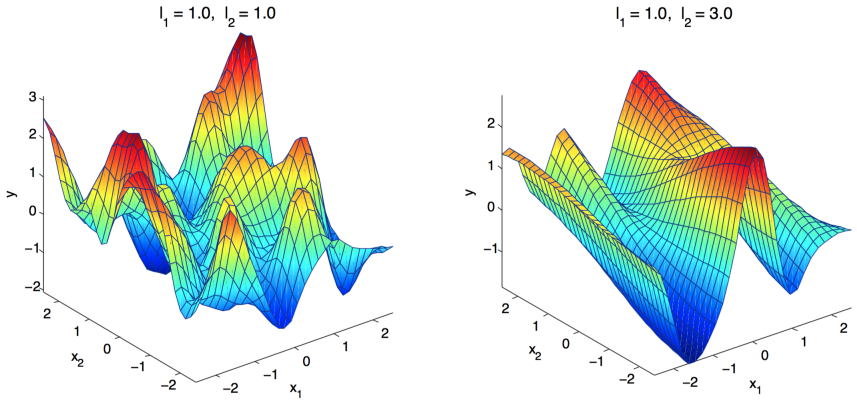

Figure 6: Illustration of GP priors using an ARD kernel with length-scale parameters ℓ_1 and ℓ_2.

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.[5]

5 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 **??**)

Exercise 5.1. Practice Retrieving!
For this summary exercise, it is intended that your answers are based on **your own** (current) understanding of the concepts (and not on the definitions you read and copy from these notes or from elsewhere). Don't hesitate to **say it out loud** to your seat neighbor, your pet or stuffed animal, or to yourself before **writing it down**. After writing it down, check your answers with your (lecture)notes and the provided reading. Correct any mistakes you made. Research studies show that this practice of retrieval and phrasing out loud will help you retain the knowledge!

(a) Using *your own words*, recap each of the above summary points in 2-3 sentences by retrieving the knowledge from the top of your head.

(b) What is the *magic of GPs*?

And always remember: It's not bad to get it wrong. *Getting it wrong is part of learning!* Use your notes or other resources to get the correct answer or come to our office hours to get help!

Extensions:

- run time $O(n^3) \rightarrow$ matrix inversion (gets slow when n is large) $\Rightarrow$ use **sparse GPs** for large n

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

- GPs are a little harder to realize for classification (we use the *logistic likelihood* which is non-Gaussian), since there is no analytic solution for the predictive distribution.
- we can model non-Gaussian likelihoods for regression, e.g., for count data (*Poisson likelihood*)
- non-Gaussian likelihoods or priors require **approximate inference techniques**

Appendix: Multivariate Gaussians

We 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:

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

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:

$$\int_{\mathbf{x}} p(\mathbf{x}; \boldsymbol{\mu}, \Sigma) \mathrm{d}\mathbf{x} = 1 \tag{A.2}$$

(2) Summation:

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

$$\mathbf{y} + \mathbf{y}' \sim \mathcal{N}(\boldsymbol{\mu} + \boldsymbol{\mu}', \Sigma + \Sigma') \tag{A.3}$$

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:

$$\mathbf{y}_A \sim \mathcal{N}(\boldsymbol{\mu}_A, \Sigma_{AA}) \tag{A.4}$$
$$\mathbf{y}_B \sim \mathcal{N}(\boldsymbol{\mu}_B, \Sigma_{BB}) \tag{A.5}$$

(4) Conditioning:

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

$$p(\mathbf{y}_A \mid \mathbf{y}_B) = \frac{p(\mathbf{y}_A, \mathbf{y}_B; \boldsymbol{\mu}, \Sigma)}{\int_{\mathbf{y}_A} p(\mathbf{y}_A, \mathbf{y}_B; \boldsymbol{\mu}, \Sigma) \mathrm{d}\mathbf{y}_A} \tag{A.6}$$

is also Gaussian:

$$\mathbf{y}_A \mid \mathbf{y}_B \sim \mathcal{N}\left(\boldsymbol{\mu}_A + \Sigma_{AB}\Sigma_{BB}^{-1}(\mathbf{y}_B - \boldsymbol{\mu}_B), \Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}\right) \tag{A.7}$$

This property is useful in deriving Gaussian Process predictions.