C24: Advanced Probability

(Michaelmas 2019)

Lecturer: Jan-Peter Calliess
Course design, exam: Michael Osborne

Course website & email: see www.robots.ox.ac.uk/~jan
Materials will also be accessible on weblearn
Topic 2: Gaussian processes

PART B: Model selection & practical considerations
Recap

Last time:

- Derived Gaussian process models over finite and infinite-dimensional function spaces.
- Derived mean and covariance functions (prior’s and posteriors).
- Derived GPs as limits of Bayesian models over finite mixtures of orthonormal function systems.
- Also, gave criterion on when a function $k(\cdot, \cdot)$ is a valid covariance function (check if pos.def.).
- Saw example of closed-form covariance function that gives rise to smooth functions.
Definitions and characterisation of covariance functions

Functions $\phi : \mathcal{X}^2 \rightarrow \mathbb{R}$ are called kernels. A covariance function $k : \mathcal{X}^2 \rightarrow \mathbb{R}$ is a kernel for which the following holds:

1. $k(x, x') = k(x', x), \ \forall x, x' \in \mathcal{X}$. (symmetric)
2. $\int_{\mathcal{X}} \int_{\mathcal{X}} f(x)k(x, x')f(x')dx dx' > 0, \ \forall f$ (pos.def. / PD)

In other words, a covariance function is a PD, symmetric kernel.

- Covariance functions are kernels for which the Gram matrices

$$C_n := ((k(x_i, x_j)))_{i,j=1,\ldots,n}$$

are pos.def. for all inputs and sizes $n \in \mathbb{N}$. Those are matrices whose eigenvalues are all positive. This is an alternative condition to (2) you can check to verify your kernel is a valid covariance function.

- Often, the PD requirement is relaxed to pos.-semi def. (PSD). That is, it suffices to check that $\nu^\top C_n \nu \geq 0$ instead of $> 0$. PSD kernels give rise to covariances that induce degenerate Gaussians that have all probability mass on a null set (the non-zero eigenvector space).
Which could be a covariance function?

Assuming \( x, x' \in [0, 1) \subset \mathbb{R} \)

1. \( K(x, x') = 1 - |x - x'| \).
2. \( K(x, x') = \mathcal{N}(x; \mu, x') \).
3. \( K(x, x') = \mathcal{P}o(x; x') \).
4. \( K(x, x') = 1 - 2xx' \).
Which could be a covariance function?

1. \( K(x, x') = 1 - |x - x'|. \)
   This is an example of a spline covariance, but that’s not so important right now.

2. \( K(x, x') = \mathcal{N}(x; \mu, x'). \)
   This function is not symmetric in \( x \) and \( x' \), meaning that the covariance matrix will not be symmetric.

3. \( K(x, x') = \mathcal{P}_o(x; x'). \)
   \( \mathcal{P}_o(x; x') = \frac{\mu^x}{x!} e^{-\mu} \) is not symmetric in \( x \) and \( x' \), meaning that the covariance matrix will not be symmetric.

4. \( K(x, x') = 1 - 2xx'. \)
   For sufficiently large \( x \), \( K(x, x) \), which must be a variance, will be negative. Hence the covariance matrix will not be positive semi-definite.
Today:

- Discuss a number of existing classes of GPs (mean and covariance functions) and the behavior of the function spaces they encode Bayesian beliefs over.
- Pragmatic data Engineering approach to model class selection: pick GP priors with means and covariances from arsenal of known ones that best matches one’s belief over the properties of the function to be learned.
- Within the class: model selection via hyperparameter optimisation.
- Some practical considerations: speeding up the inference and applications.
The prior mean function $\mu(x)$ should be our best guess (of any form) for the function $y(x)$ before any observations are made.
Remember:

\[
[K(x_*, \xi_1), \ldots, K(x_*, \xi_n)]
\]

call this

\[
a = [a_1, \ldots, a_n]^	op
\]

\[
m_{*|d} = \mu(x_*) + K(x_*, x_d) \left( K(x_d, x_d) + \sigma^2 I_d \right)^{-1} (y_d - \mu(x_d))
\]

\[
= \mu(x_*) + \sum_{j=1}^{n} a_j K(x_*, \xi_j)
\]

Two observations:

1) The complexity and computational effort for evaluating the posterior mean function grows with the data size n. Such regression methods are called **nonparametric**. (in contrast, in parametric methods the prediction complexity is fully dependent on a fixed-size parameter).
2) Typically: $K(x, x') \to 0$ as distance $d(x, x')$ between inputs $x, x'$ grows.

\[
m_{\star|d} = \mu(x_{\star}) + \sum_{j=1}^{n} a_{j} K(x_{\star}, \xi_{j})
\]

Hence:

The prior mean function is the function our inference will default to far from observations, important for extrapolation.
Test points far from observations are an important consideration for high-dimensional data.
Some pragmatic heuristics for picking a prior mean function:

<table>
<thead>
<tr>
<th>Conditions</th>
<th>Mean function ($\mu$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exceptionally lazy</td>
<td>0</td>
</tr>
<tr>
<td>Ordinarily lazy</td>
<td>$\text{mean}(y_d)$</td>
</tr>
<tr>
<td>Interested only in interpolation</td>
<td>constant, $\theta$</td>
</tr>
<tr>
<td>Extrapolation required</td>
<td>Bespoke parametric model built using domain knowledge</td>
</tr>
</tbody>
</table>
Covariance functions allow us to incorporate structure and correlations into our models.

- **Periodicity**
- **Delays between sensors**
- **Long-term drifts**
- **Correlated sensors**
Many common covariance functions express that covariance decreases with increasing Euclidean distance:

\[ K(x, x'; \theta) = \lambda^2 f \left( \frac{d(x, x')}{\omega} \right), \]

where \( f \) is a monotonically decreasing function, and \( d \) is the Euclidean distance,

\[ d(x, x') = \sqrt{\sum_i (x_i - x'_i)^2}. \]

\[ k(x, x') = K(x, x'; \theta) \quad : \text{cov. function for any hyperparameter choice} \quad \theta = (\lambda, \omega) \]

can be non-Euclidean too!
The hyperparameters $\lambda > 0$ and $\omega > 0$ specify our expected length scales of the function in output ('height') and input ('width') spaces respectively.
The exponentiated quadratic covariance is useful for smooth data:

\[ K(x, x'; \theta) = \lambda^2 \exp -\frac{1}{2} \left( \frac{d(x, x')}{\omega} \right)^2. \]
Use the **rational quadratic** covariance for data that is smooth across a range of length scales:

\[
K(x, x'; \theta) = \lambda^2 \left( 1 + \frac{1}{2\alpha} \left( \frac{d(x, x')}{\omega} \right)^2 \right)^{-\alpha}.
\]
The Matérn class of covariances are useful for functions of variable smoothness:

\[ K(x, x'; \theta, \nu = \frac{1}{2}) = \lambda^2 \exp\left(-\frac{d(x, x')}{\omega}\right); \]

\[ K(x, x'; \theta, \nu = \frac{3}{2}) = \lambda^2 \left(1 + \sqrt{3} \frac{d(x, x')}{\omega}\right) \exp\left(-\sqrt{3} \frac{d(x, x')}{\omega}\right). \]
The periodic covariance is useful for periodic data:

\[ K(x, x'; \theta) = \lambda^2 \exp \left( -\frac{2\left( \sin(\pi d(x, x')/\rho) \right)^2}{\omega} \right); \]

where \( \rho \) is the period and \( \omega \) controls the roughness.
The **affine** covariance exactly corresponds to an affine model with Gaussian priors:

\[ K(x, x'; \theta) = \alpha^2 + \beta^2(x - \gamma)(x' - \gamma); \]

recall our derivation at the start of this topic.
**Theorem**

The class of kernels that are covariance functions is closed under addition, multiplication and positive scaling:

Let $k_1, k_2 : \mathcal{X}^2 \to \mathbb{R}$ be two PD (PSD) and symmetric kernel functions. Then we have:

- $k_1 + k_2$ is PD (PSD) and symmetric,
- $k_1 k_2$ is PD (PSD) and symmetric and
- $\forall r > 0 : r k_1$ is PD (PSD) and symmetric.

This allows us to synthesise new covariance functions as a combination of ones that we already know...
We can create new covariance functions by adding and/or multiplying other covariance functions.

\[ \times \times ( + ) \]

**Multiplication (and)** corresponds to covariance being dependent on similarity under both terms.

**Addition (or)** corresponds to covariance being dependent on similarity under either term.
When a function is the sum of two independent functions:

use a covariance that is the sum of the covariances for those two functions,

\[ K_a(x, x') + K_b(x, x') \].

Adding kernels

Roughly speaking, adding two kernels can be thought of as an OR operation. That is, if you add together two kernels, then the resulting kernel will have high value if either of the two base kernels have a high value.

Quote from https://www.cs.toronto.edu/~duvenaud/cookbook/
When a function is the **product** of two independent functions:

use a covariance that is the product of the covariances for those two functions,

\[ K_a(x, x') K_b(x, x') \]

E.g.:

A linear kernel times a periodic results in functions which are periodic with increasing amplitude as we move away from the origin.

**Multiplying Kernels**

Multiplying together kernels is the standard way to combine two kernels, especially if they are defined on different inputs to your function. Roughly speaking, multiplying two kernels can be thought of as an AND operation. That is, if you multiply together two kernels, then the resulting kernel will have high value only if both of the two base kernels have a high value.

Quote from https://www.cs.toronto.edu/~duvenaud/cookbook/
The linear plus periodic covariance:

Draws

\[ K(0, x) \]

\[ f(x) \]
The linear times linear covariance:

Draws

... a quadratic... of course, can form any polynomial combination of kernels...
The linear times exponentiated quadratic covariance.
The periodic times exponentiated quadratic covariance:
Which covariance function might you use to model historical fur sales in the US?

Plot courtesy of David Duvenaud.
Which covariance function should you choose?

1. Rational quadratic times periodic.
2. Exponentiated quadratic.
3. Exponentiated quadratic plus periodic.
4. Matérn, $\nu = \frac{1}{2}$.
Which covariance function should you choose?
NB: there is no ‘right’ answer!

1. Rational quadratic times periodic.
2. Exponentiated quadratic.
3. Exponentiated quadratic plus periodic.
4. Matérn, $\nu = \frac{1}{2}$. 
We can create covariances for functions of many dimensions.

1. We can simply use the Euclidean distance in $\mathbb{R}^n$;
2. Alternatively, we can take a covariance that is the product or sum of covariances, one for each dimension:

$$K_1(x_1, x'_1) K_2(x_2, x'_2) \quad \text{or} \quad K_1(x_1, x'_1) + K_2(x_2, x'_2).$$
We tackle **multiple outputs** with co-regionalisation.

If the $l$th output is $f_l(x)$, rewrite as $g(l, x) = f_l(x)$. Then we can take a covariance for $g$ of the form $K_L(l, l') K_X(x, x')$.

Note that if there are 3 outputs, $K_L$ is specified by a 3 by 3 positive semi-definite matrix.
Hyperparameters can have a significant influence on inference.

The input scale for $A$ is short, longer for $B$ and longest for $C$. 

Model selection

- A GP is a Bayesian model. Typically, the parameters can be construed as the (potentially infinitely long) weights $w = (w_1, w_2, \ldots)$ in our mixture model $\sum_i w_i \psi_i$.

- Remember, we performed model selection by evidence maximisation. Here, we might pick:

$$\mathcal{M} \in \arg \max_{\mathcal{M}} p(\mathcal{D}|\mathcal{M}) = \int p(\mathcal{D}|w, \mathcal{M}) p(w|\mathcal{M}) \, dw.$$
Model selection

- GP prior is implicitly given by the mean and covariance functions. Assume the prior mean is fixed and that we commit to a class of covariances

\[ \{ K(\cdot, \cdot; \theta) | \theta \in \Theta \} \]

with hyperparameters in some space \( \Theta \).

- Then, the model selection translates to hyperparameter optimisation: Choose \( \mathcal{M}_{\theta^*} \) such that:

\[
\theta^* \in \operatorname{arg} \max_{\theta} p(\mathcal{D}|\mathcal{M}_\theta) = \int p(\mathcal{D}|w, \mathcal{M}_\theta) p(w|\mathcal{M}_\theta) \, dw.
\]

- Alternatives to this Maximum Likelihood optimisation approach are MAP (max. a posteriori) optimisation or even marginalisation after placing a meta-prior \( p(\theta) \) over the hyperparameters.
Maximum likelihood and MAP are popular approaches to setting hyperparameters.

The GP log-likelihood is (where $n_d$ is the number of data $y_d$)

$$p(D|M_\theta)$$

$$\log p(y_d | \theta) = -\frac{1}{2} (y_d - \mu(x_d))^\top (K(x_d, x_d) + \sigma^2 I_d)^{-1} (y_d - \mu(x_d))$$

penalises mismatch between prior and data

$$-\frac{1}{2} \log \det(K(x_d, x_d) + \sigma^2 I_d) - \frac{n_d}{2} \log 2\pi;$$

penalises model complexity

constant

$\mu$ and $K$ are both functions of $\theta$. Note that many of our hyperparameters $\theta$ (e.g. length scales $\omega$ and $\lambda$) are strictly positive: it is common to reparameterise to optimise over the logarithms e.g. $\tilde{\omega} = \log \omega$. 

37
The first term in the log-likelihood will deform the prior to match the data.

The first term is the log-density of (an un-normalised version of) the Gaussian prior evaluated at the data,

$$\frac{1}{2} (y_d - \mu(x_d))^\top \left( K(x_d, x_d) + \sigma^2 I_d \right)^{-1} (y_d - \mu(x_d)).$$
The second term in the log-likelihood will penalise excessive complexity: the ability to explain too much data.

Let’s write use the eigenvector decomposition

\[ K(x_d, x_d) + \sigma^2 I_d = RR^\top \]

where \( R \) is an orthogonal (rotation) matrix and \( \Lambda \) is the diagonal eigenvalue matrix. The second term in the log-likelihood can hence be written as

\[ -\frac{1}{2} \log \det(K(x_d, x_d) + \sigma^2 I_d) = -\frac{1}{2} \log \prod_i \Lambda_{i,i}. \]

That is, this term is largest when the product of eigenvalues of the covariance matrix is smallest.
The second term in the log-likelihood will penalise excessive complexity.
This term is largest when the product of eigenvalues of the covariance matrix is smallest: when the model is compatible with very few datasets.

Product of eigenvalues small  ⇒  likelihood term high
Product of eigenvalues large  ⇒  likelihood term low
Optimising the likelihood will push the prior mean $\mu$ to fit the data $y_d$ exactly.

The Gaussian process is bolted onto a parametric model: the prior mean. Doing maximum likelihood for a parametric model will lead to exactly this kind of over-fitting, as we know! We get around this in practice by taking only very simple prior mean functions that can't do much damage if we use maximum likelihood.

(or not optimising prior mean hyperparameters at all)
The key computational bottleneck associated with Gaussian processes is resolving $K^{-1}v$.

Solving such equations crops up in many places:

$$m_{*|d} = \mu(x_*) + K(x_*, x_d) \left( K(x_d, x_d) + \sigma^2 I_d \right)^{-1} (y_d - \mu(x_d));$$

$$C_{*|d} = K(x_*, x_*) - K(x_*, x_d) \left( K(x_d, x_d) + \sigma^2 I_d \right)^{-1} K(x_d, x_*);$$

$$\log p(y_d | \theta) = -\frac{1}{2} \left( y_d - \mu(x_d) \right)^\top \left( K(x_d, x_d) + \sigma^2 I_d \right)^{-1} \left( y_d - \mu(x_d) \right)$$

$$- \frac{1}{2} \log \det \left( K(x_d, x_d) + \sigma^2 I_d \right) - \frac{n_d}{2} \log 2\pi.$$
Note that computing $K^{-1}v$ is equivalent to solving $v = Ku$ for $u$.

You should never explicitly compute a matrix inverse.

Matrix inversion is $\mathcal{O}(n_d^3)$ for a matrix of size $n_d$.

Worse, matrix inversion is relatively unstable and susceptible to conditioning errors.
Conditioning issues emerge when some of your data is excessively self-similar, leading to high covariances.

\[
\begin{bmatrix}
K(x_1, x_1) & K(x_1, x_2) & K(x_1, x_3) & \cdots \\
K(x_2, x_1) & K(x_2, x_2) & K(x_2, x_3) & \cdots \\
K(x_3, x_1) & K(x_3, x_2) & K(x_3, x_3) & \cdots \\
\vdots & \vdots & \vdots & \ddots \\
1.00000 & 0.99999 & 0.99998 & \cdots \\
0.99999 & 1.00000 & 0.99999 & \cdots \\
0.99998 & 0.99999 & 1.00000 & \cdots \\
\vdots & \vdots & \vdots & \ddots 
\end{bmatrix}
\]

where the first two rows might be identical to four significant figures.
Conditioning is ameliorated by noise.

\[ K(x_d, x_d) + \sigma^2 I_d \]

\[
\begin{bmatrix}
K(x_1, x_1) + \sigma^2 & K(x_1, x_2) & K(x_1, x_3) & \cdots \\
K(x_2, x_1) & K(x_2, x_2) + \sigma^2 & K(x_2, x_3) & \cdots \\
K(x_3, x_1) & K(x_3, x_2) & K(x_3, x_3) + \sigma^2 & \cdots \\
\vdots & \vdots & \vdots & \ddots
\end{bmatrix}
\]

For this reason, we often assume noise where there isn’t, or add on a small amount to \( \sigma \) (known as jitter), so as to combat conditioning.
As jitter is, essentially, artificial noise, it artificially dilutes the informativeness of data.
The **Cholesky** factorisation of a positive semi-definite matrix is faster than inversion \( \frac{1}{3} \mathcal{O}(n_d^3) \) and is more numerically stable.

\[
K(x_d, x_d) + \sigma^2 I_d = R^\top R
\]

where

\[
R = \text{chol}(K(x_d, x_d) + \sigma^2 I_d)
\]

\[
= \begin{bmatrix}
R_{11} & R_{12} & R_{13} & \cdots \\
0 & R_{22} & R_{23} & \cdots \\
0 & 0 & R_{33} & \cdots \\
\vdots & \vdots & \vdots & \ddots
\end{bmatrix}
\]
The Cholesky factor $R$ can be used to solve $v = Ku$ using $O(n_d^2)$ back-substitution.

Given $v$ and $R$ (where $K = R^\top R$), we find $u$ as

$$u = K^{-1}v$$

$$\Rightarrow v = K u$$

and

$$u' = Ru$$

Thus,

$$
\begin{bmatrix}
v_1 \\
v_2 \\
v_3 \\
\vdots \\
u'_1 \\
u'_2 \\
u'_3 \\
\vdots
\end{bmatrix}
= 
\begin{bmatrix}
R_{11} & 0 & 0 & \cdots \\
R_{12} & R_{22} & 0 & \cdots \\
R_{13} & R_{23} & R_{33} & \cdots \\
\vdots & \vdots & \vdots & \ddots \\
R_{11} & R_{12} & R_{13} & \cdots \\
0 & R_{22} & R_{23} & \cdots \\
0 & 0 & R_{33} & \cdots \\
\vdots & \vdots & \vdots & \ddots
\end{bmatrix}
\begin{bmatrix}
u'_1 \\
u'_2 \\
u'_3 \\
\vdots
\end{bmatrix}
$$
The Cholesky factor can also be used to compute the determinant.

Recall

\[
\log p(y_d | \theta) = -\frac{1}{2} (y_d - \mu(x_d))^\top (K(x_d, x_d) + \sigma^2 I_d)^{-1} (y_d - \mu(x_d)) \\
- \frac{1}{2} \log \det(K(x_d, x_d) + \sigma^2 I_d) - \frac{n_d}{2} \log 2\pi.
\]

Now

\[
\log \det R^\top R = \log(\det R^\top \det R) \\
= \log(\det R \det R) \\
= 2 \log \det R \\
= 2 \log \prod_i R_{ii} \\
= 2 \sum_i \log R_{ii}.
\]
In summary,

1. The **Gaussian process** is a flexible, non-parametric, distribution for functions.
2. The **mean function** controls extrapolation.
3. The **covariance function** encodes structure about the function.
4. Covariance functions can be **combined** and **modified**.
5. **Maximum likelihood** can be used for Gaussian process hyperparameters.
6. **Cholesky factorisation** helps with the computational challenges for Gaussian processes.

(numerical)
Appendix

—- Some additional content (for those who are interested)
Appendix - Some things that are nice to know….

Eigenvalues and eigenfunctions of the RKHS Operator

- Kernels got their name from being utilised as integral kernels of (RKHS) operators. For a kernel $k$ its RKHS OP $T_k : \mathcal{F} \to \mathcal{F}$, $\mathcal{F} \subseteq L_2 \cap L_1$ maps a function $f$ to a function $T_k f$ with

$$T_k f(x) := \int k(x, x') f(x) dx.$$ 

- Now, our requirement for a kernel to be a covariance function means that $\langle f, T_k f \rangle_{L_2} > 0 \forall f$.

- Similarly, to finite-dim. linear algebra, this PD condition entails that all Eigenvalue are positive: $\text{spec}(T_k) \subset \mathbb{R}_+$.

- Together with symmetry, this implies a discrete spectrum of eigenvalues.
Eigenvalues and eigenfunctions of the RKHS Operator

• Now, assume that our kernel was constructed via our Bayesian model obtained from imposing priors over the weights \( w_i (i \in \mathbb{N}) \) in the mixture model \( \sum_{i \in \mathbb{N}} w_i \psi_i \). Furthermore, remember that \( k(x, x') = \sum_i v_i \psi_i(x) \psi_i(x') \) with \( v_i = \text{var}(w_i) \).

• If the \( \psi_i \) are an ONS then \( \langle \psi_i(\cdot), \psi_j(\cdot) \rangle_{L_2} = \begin{cases} 1, & i = j \\ 0, & i \neq j. \end{cases} \) This implies,

\[
T_k \psi_j(x) = \langle k(x, \cdot), \psi_j \rangle_{L_2} = \langle \sum_i v_i \psi_i(x) \psi_i(\cdot), \psi_j(\cdot) \rangle_{L_2} = \langle \sum_i v_i \psi_i(x) \psi_i(\cdot), \psi_j(\cdot) \rangle_{L_2} = \sum_i v_i \psi_i(x) \langle \psi_i(\cdot), \psi_j(\cdot) \rangle_{L_2} = v_j \psi_j(x) 1 = v_j \psi_j(x). \tag{9}
\]

• \( \rightarrow \psi_i \) are the eigenfunctions of \( T_k \) with pertaining eigenvalues \( \text{var}(w_i) \). So the PD (PSD) requirement of covariance functions essentially means that the \( \text{var}(w_i) > 0(\geq 0) \) as they should be!
Recall Gaussian distributed variables are joint Gaussian with any affine transform of them; this includes rotations, scalings and translations.

This means we can use observations of e.g. \( ax_1 + bx_2 + c \) to infer e.g. \( x_1 \) in closed form.
A function over which we have a Gaussian process is joint Gaussian with any integral or derivative of it, as integration and differentiation are affine.
Hence we can include derivative observations into our Gaussian process.
We can also use observations of an integrand $\ell$ to perform inference for its integral, $Z$: this is known as **Bayesian quadrature**.
Maximum likelihood and MAP are popular approaches to setting hyperparameters.

Aren’t maximum likelihood and MAP bad? Yes!

However, here we are using them for hyperparameters:

our parameters (weights w) are properly marginalised!

The further up the hyper-chain you go, the less influence your approximations should have: let’s justify the use of maximum likelihood for this setting.

Solution: could introduce meta-prior over hyperparameters....
We need to approximate the usual ratio of integrals:

\[ p(f_* \mid y_d) = \frac{\int p(f_* \mid y_d, \theta) p(y_d \mid \theta) \, d\theta \, p(\theta)}{\int p(y_d \mid \theta) \, p(\theta) \, d\theta}, \]

where \( p(f_* \mid y_d, \theta) \) are the predictions, \( p(y_d \mid \theta) \) is the likelihood and \( p(\theta) \) is the prior.

Maximum likelihood and maximum a-posteriori (MAP) approximate these integrals be approximating the likelihood \( p(y_d \mid \theta) \) and posterior (proportional to \( p(y_d \mid \theta) \, p(\theta) \)), respectively, as delta functions of \( \theta \).