Model Selection for Gaussian Process Regression by Approximation Set Coding

MODEL SELECTION FOR GAUSSIAN PROCESS REGRESSION BY APPROXIMATION SET CODING

A master's thesis by

BENJAMIN FISCHER

with supervisor

PROFESSOR JOACHIM M. BUHMANN

and advisers

YATAO BIAN
STEFAN BAUER

DEPARTMENT OF COMPUTER SCIENCE
ETH ZURICH
MAY 9, 2016

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Abstract

Gaussian processes are powerful, yet analytically tractable models for supervised learning. As a generalization of the multivariate Gaussian distribution, a Gaussian process is characterized by a mean function and a covariance function. The problem of model selection is to determine a mean and covariance function with the aim of adapting the Gaussian process to given data points – a difficult balancing act between data fit and model complexity. The functions to be compared are in essence arbitrary, since they do not just differ in their parametrization but in their fundamental structure. In domains such as systems biology it is often not clear which function structure to choose, for instance to decide between a squared exponential and a rational quadratic covariance function. Based on the theory of approximation set coding (ASC), a framework for model selection is developed, which is general enough to do hyperparameter optimization for any model that has a prior on its parameters. The framework is then applied to Gaussian process regression. Experiments on synthetic and real-world data are presented to compare approximation set coding to the classic model selection criteria of maximum evidence (also known as marginal likelihood) and leave-one-out cross-validation. Although approximation set coding shows promise to become a competitive model selection criterion, it currently seems not to perform better than the classic criteria in our experiments. Maximum evidence has the best performance in general, while approximation set coding occasionally surpasses leave-one-out cross-validation. Further work is needed on systematic ways to compare model selection criteria, or even combine them in ensembles.

Acknowledgments

My advisers Yatao Bian and Stefan Bauer deserve a lot of gratitude for their continuous inspiration. Our discussions made me look at problems from different points of view, which was crucial to overcome unclear phases of the project. This thesis would not be the same without Nico S. Gorbach. He took all the time in the world to explain things, with his help being of key importance during the whole process. Lastly, I would like acknowledge how my supervisor Professor Joachim M. Buhmann assisted me with his brainchild at the basis of this work. Thanks.

Contents

Naming Conventions

The columns of a matrix $\mathbf{A}$ are denoted by $\mathbf{a}_n$ and its entries by $a_{d,n}$. For a vector $\mathbf{x}$, its dth entry is denoted by x_d.

Name	Meaning
$\mathcal{C}$	Set of all hypotheses in approximation set coding
$(\cdot)_{CV}$	Related to the criterion of cross-validation
$\mathbf{c}$	Hypothesis in approximation set coding
D	Number of dimensions of a data set
$\mathcal{D}$	Data set; in case of regression, $\mathcal{D} = \{(\mathbf{x}_n, y_n) \mid n = 1, \ldots, N\}$
$F(\mathbf{x})$	Cumulative distribution function
$\mathbf{f}$	Vector of Gaussian process latent function values
$\mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$	Gaussian process with the mean function m and the covariance function k
$\mathcal{H}$	Discrete set of possible functional forms (model structures)
I_β^θ	Mutual information in approximation set coding
J	Number of random partitions to estimate the approximation capacity in approximation set coding
K	Number of subsets in K-fold cross-validation
$\mathbf{K}$	Gaussian process covariance matrix in general
$\mathbf{K}_f$	Gaussian process covariance matrix for the noise-free $\mathbf{f}$
$\mathbf{K}_y$	Gaussian process covariance matrix for the noisy $\mathbf{y}$
$k(\mathbf{x}, \mathbf{x}')$	Gaussian process covariance function
L	Number of instances in approximation set coding
$\mathbf{L}$	Lower triangular matrix of the Cholesky decomposition
$(\cdot)_l$	Related to the lth subset in approximation set coding
ℓ	Characteristic length-scale hyperparameter of several covariance functions
M	Number of objects in approximation set coding
$(\cdot)_{ME}$	Related to the criterion of maximum evidence (also known as marginal likelihood)
m	History size of the limited-memory BFGS optimization algorithm [27]
$m(\mathbf{x})$	Gaussian process mean function
$\mathbf{m}$	Gaussian process mean vector
N	Number of cases in a data set

Name	Meaning
$p(\mathbf{x})$	Probability density function
$R(\mathbf{c}, \mathcal{D})$	General cost function for a hypothesis given data
$R_2(\mathbf{c}, \mathcal{D})$	Cost function for hyperparameter optimization
$R_3(\mathbf{c}, \mathcal{D})$	Cost function for functional form selection
T	Period hyperparameter of the periodic covariance function of Equation (2.8) on page 14
$w_\beta(\mathbf{c}, \mathcal{D})$	Weight function in approximation set coding
$\mathbf{X}$	Matrix of regression inputs
$\mathbf{y}$	Vector of regression outputs
α	Shape hyperparameter of the rational quadratic covariance function of Equation (2.6) on page 14
$\boldsymbol{\alpha}$	Parameter vector of a model such as linear regression
β	Approximation precision in approximation set coding
$\boldsymbol{\epsilon}$	Noise vector
η_β	Posterior agreement in approximation set coding
θ	Single hyperparameter of a model
$\boldsymbol{\theta}$	Hyperparameter vector of a model
$\boldsymbol{\mu}_0$	Prior mean vector
$\boldsymbol{\Sigma}_0$	Prior covariance matrix
σ_0^2	Prior variance
σ_f^2	Variance of the noise-free signal f
σ_n^2	Variance of the Gaussian noise added to f
τ	Transformation in approximation set coding

1. Introduction

A Gaussian process defines a distribution over functions. One would like to adapt this distribution to a given set of data points, for example to explore periodicity of the data. Model selection is to solve this problem of adapting to data. Selecting a Gaussian process model essentially means to specify a mean function and a covariance function.

As one can imagine, selecting a function is a hard problem, since the options are virtually unlimited. Typically, one considers a handful of parametrized function classes, each of which has gaps to be filled with numbers in order to produce an actual function. These gaps are the hyperparameters, while we refer to a function class as a functional form. Table 1.1 on the facing page gives an intuition about how Gaussian processes look like for various functional forms and hyperparameters, adapted to sample data points. From the Gaussian processes of this example (and perhaps infinitely many more), which one should we select? That is the problem of model selection. In domains such as systems biology [37], there is often no prior knowledge for selecting a certain functional form.

Two well-known options to do model selection for Gaussian processes exist. The criterion of maximum evidence (also known as marginal likelihood) maximizes the probability of the data under the model assumptions. Cross-validation, on the other hand, minimizes an estimated generalization error of the model. Under certain circumstances, maximum evidence is less resistant to model misspecification, whereas cross-validation can suffer from a higher variance [2]. There is a lack of empirical studies about which of these two criteria to use under which circumstances [30]. Going one step further, it is an open problem how to combine the strengths of maximum evidence and cross-validation into a unifying model selection criterion.

1.1. Contribution

Approximation set coding (ASC) is an abstract theory to determine an optimal trade-off between the expressiveness of a model and the reproducibility of its inference [7]. For instance, as a model selection principle applied to data clustering, it is able to compare clustering models and choose the number of clusters [10]. A first contribution of this work is to transfer approximation set coding to any models that define a parameter prior and a likelihood, as is the case for Bayesian linear regression. This gives birth to a family of algorithms to do model selection on the level of hyperparameters.

Second, the developed framework is instantiated for Gaussian process regression, which naturally has a prior and a likelihood. The resulting model selection criterion is

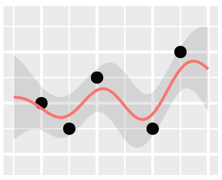
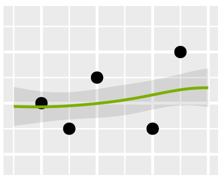
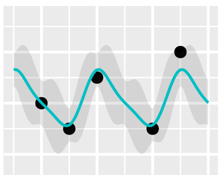
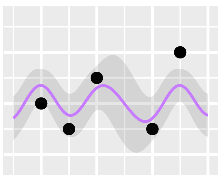
Functional form	Hyperparameters	Gaussian process
Squared exponential covariance function	$\ell = 1$ $\sigma_f = 2$	
Squared exponential covariance function	$\ell = 2$ $\sigma_f = 1$	
Periodic covariance function	$\ell = 1$ $T = 3$ $\sigma_f = 2$	
Periodic covariance function	$\ell = 1$ $T = 5$ $\sigma_f = 2$	

Table 1.1.: Model selection for Gaussian processes means to select a functional form and its hyperparameters. For given data points, the selected Gaussian process can then be viewed as a distribution over functions. The mean of this distribution is visualized, plus and minus one standard deviation.

then compared to the classics of maximum evidence and leave-one-out cross-validation, for both hyperparameter optimization and functional form selection. Although the experiments do not suggest that approximation set coding performs better in general, there are cases where it surpasses leave-one-out cross-validation. In one scenario for hyperparameter optimization based on synthetic data, approximation set coding has a similar median, but a lower variance than leave-one-out cross-validation, with respect to the standardized mean squared error. Maximum evidence is mostly the top performer. In another scenario for functional form selection based on real-world data, it is interesting to see how maximum evidence and leave-one-out cross-validation disagree on which covariance function should suit the data best.

The implications of this work are twofold. Approximation set coding has potential to be an alternative model selection criterion for Gaussian process regression, but still needs improvement in future work to be competitive with maximum evidence and cross-validation. In addition, it seems advisable to always try more than one criterion, but the question about how to combine them remains unanswered.

2. Background

In the following, we summarize relevant literature to set the stage. Regarding Gaussian processes, Rasmussen and Williams [30] give a comprehensive treatment of the subject in the context of machine learning.

2.1. Gaussian Processes

The multivariate Gaussian distribution is a generalization of the one-dimensional Gaussian distribution from a distribution over scalars to a distribution over vectors. Similarly, a Gaussian process is a generalization of the multivariate Gaussian distribution from a distribution over vectors to a distribution over functions. See Table 2.1 for these levels of abstraction. Informally speaking, a function $f : \mathbb{R}^D \to \mathbb{R}$ can be seen as an infinitely long vector [30]: given an argument $x \in \mathbb{R}^D$ acting as a vector index, the function f produces the value $f(x)$ acting as a vector entry.

More formally, a Gaussian process is characterized by a mean function $m : \mathbb{R}^D \to \mathbb{R}$ and a covariance function $k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$. We denote it by $\mathcal{GP}(m(x), k(x, x'))$, which defines a distribution over functions $f : \mathbb{R}^D \to \mathbb{R}$ as follows. Given N inputs arranged as columns of a matrix $\mathbf{X} \in \mathbb{R}^{D \times N}$, we get a joint distribution over the corresponding N function values $f(x_n)$ arranged as a vector $\mathbf{f} \in \mathbb{R}^N$. The distribution over $\mathbf{f}$ is an N-dimensional Gaussian

$$\mathbf{f} \mid \mathbf{X} \sim \mathcal{N}(\mathbf{m}, \mathbf{K}_f), \tag{2.1}$$

where $m_n = m(x_n)$ and $(k_f)_{n,n'} = k(x_n, x_{n'})$. That is, the mean function defines the mean vector $\mathbf{m}$ and the covariance function the covariance matrix $\mathbf{K}_f$. To lift a mean or covariance function from vectors to matrices, we write $\mathbf{m}(\cdot)$ or $\mathbf{K}_f(\cdot, \cdot)$, respectively.

Level of abstraction	Distribution over	Notation
Gaussian distribution	Scalars	$\mathcal{N}(\mu, \sigma^2)$
Multivariate Gaussian distribution	Vectors	$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
Gaussian process	Functions	$\mathcal{GP}(m(x), k(x, x'))$

Table 2.1.: Gaussian processes as a generalization of Gaussian distributions.

2.1.1. Regression

When using Gaussian processes as a regression model, the vector $\mathbf{f}$ of latent function values is hidden. Instead, we observe the corresponding outputs $\mathbf{y} \in \mathbb{R}^N$ with independent Gaussian noise added. That is, $\mathbf{y} = \mathbf{f} + \boldsymbol{\epsilon}$ with independent and identically distributed $\epsilon_n \sim \mathcal{N}\left(0, \sigma_n^2\right)$ for $n = 1, \ldots, N$, so that the likelihood is

$$\mathbf{y} \mid \mathbf{f} \sim \mathcal{N}\left(\mathbf{f}, \sigma_n^2 \mathbf{I}\right). \tag{2.2}$$

The "n" in σ_n stands for "noise" (not for an index n), which should avoid confusion with other standard deviations later. The noise level σ_n is the first hyperparameter we encounter, with the remaining hyperparameters defining the mean and covariance function m and k, respectively. In the context of Bayesian statistics, the vector $\mathbf{f}$ can be seen as the parameters of this model with the prior $p\left(\mathbf{f} \mid \mathbf{X}\right)$ given by Equation (2.1) on page 11.

With the knowledge of both the prior and the likelihood, the evidence can be inferred using Proposition 3 on page 15 to be

$$\mathbf{y} \mid \mathbf{X} \sim \mathcal{N}\left(\mathbf{m}, \mathbf{K}_y\right), \tag{2.3}$$

where $\mathbf{K}_y = \mathbf{K}_f + \sigma_n^2 \mathbf{I}$. With this in mind, we continue to make predictions. Suppose we have already seen the noisy outputs $\mathbf{y}$ for the inputs $\mathbf{X}$. We would like to know how the noise-free function values $\widetilde{\mathbf{f}}$ are distributed for unseen inputs $\widetilde{\mathbf{X}}$. The joint distribution is

$$\begin{bmatrix} \widetilde{\mathbf{f}} \\ \mathbf{y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \widetilde{\mathbf{m}} \\ \mathbf{m} \end{bmatrix}, \begin{bmatrix} \widetilde{\mathbf{K}}_f & \mathbf{A} \\ \mathbf{A}^T & \mathbf{K}_y \end{bmatrix}\right),$$

where

$$\widetilde{\mathbf{m}} = \mathbf{m}\left(\widetilde{\mathbf{X}}\right), \qquad \widetilde{\mathbf{K}}_f = \mathbf{K}_f\left(\widetilde{\mathbf{X}}, \widetilde{\mathbf{X}}\right), \qquad \mathbf{A} = \mathbf{K}_f\left(\widetilde{\mathbf{X}}, \mathbf{X}\right).$$

Applying Proposition 4 on page 15, the predictive distribution is

$$\widetilde{\mathbf{f}} \,\Big|\, \mathbf{X}, \mathbf{y}, \widetilde{\mathbf{X}} \sim \mathcal{N}\left(\widetilde{\mathbf{m}} + \mathbf{A}\mathbf{K}_y^{-1}\left(\mathbf{y} - \mathbf{m}\right), \widetilde{\mathbf{K}}_f - \mathbf{A}\mathbf{K}_y^{-1}\mathbf{A}^T\right). \tag{2.4}$$

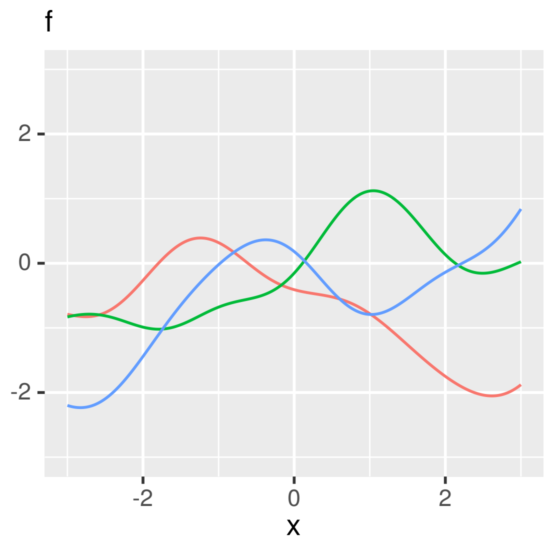
In fact, this is the posterior (another Gaussian process) evaluated at $\widetilde{\mathbf{X}}$. It allows us to make predictions associated with information about how uncertain they are.

2.1.2. Examples of Covariance Functions

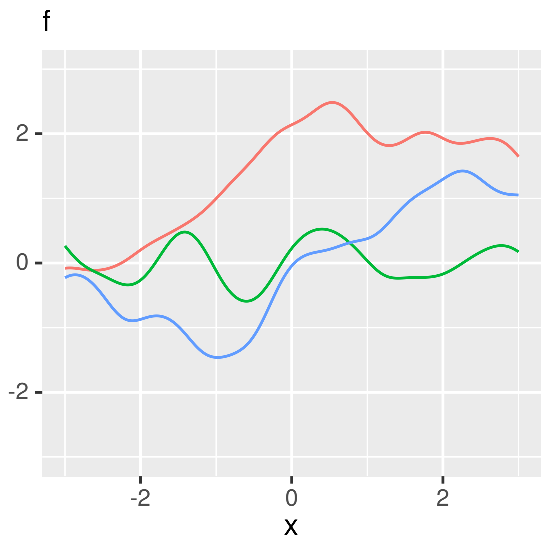
The selection of a covariance function is crucial, as it models how similar outputs are for close inputs. Figure 2.1 on the next page illustrates functions f drawn from various example covariance functions. Covariance functions can be combined. For instance, a sum or product of covariance functions is another valid covariance function.

Probably the most popular covariance function [13] is the squared exponential covariance function

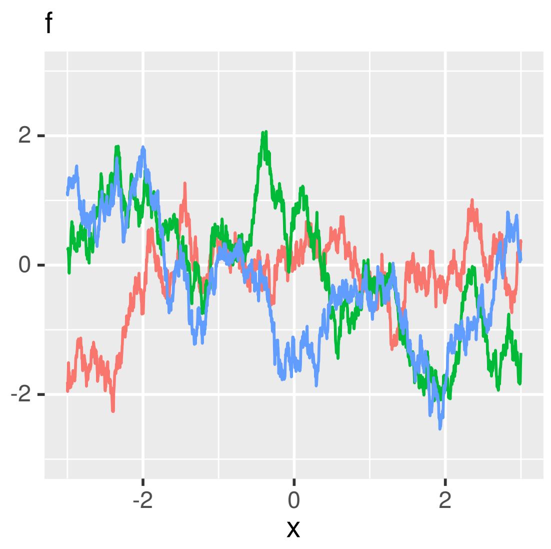
$$k\left(\mathbf{x}, \mathbf{x}'\right) = \sigma_f^2 \exp\left(-\frac{1}{2\ell^2}\left\|\mathbf{x} - \mathbf{x}'\right\|_2^2\right). \tag{2.5}$$



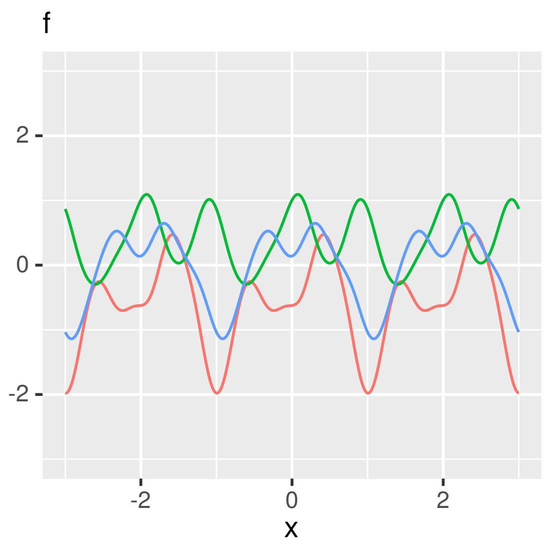
(a) Squared exponential covariance function of Equation (2.5) on page 12 where $\ell = 1$ and $\sigma_f = 1$.



(b) Rational quadratic covariance function of Equation (2.6) on the next page where $\ell = 1$, $\sigma_f = 1$, and $\alpha = 1/2$.



(c) Exponential covariance function of Equation (2.7) on the next page where $\ell = 1$ and $\sigma_f = 1$.



(d) Periodic covariance function of Equation (2.8) on the next page where $\ell = 1$, $T = 2$, and $\sigma_f = 1$.

Figure 2.1.: Sample functions f randomly drawn from Gaussian processes with the zero mean function and various covariance functions. The x-axis is discretized by 2048 equally spaced points.

It is very smooth. The hyperparameter σ_f allows us to adapt to any desired variance. The other hyperparameter ℓ is called the characteristic length-scale. For $D = 1$, a zero-mean Gaussian process crosses the x-axis in expectation $(2\pi\ell)^{-1}$ times upwards on the unit interval [30].

An infinite sum of squared exponential covariance functions with different characteristic length-scales is equivalent to the rational quadratic covariance function

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \left(1 + \frac{1}{2\alpha\ell^2} \|\mathbf{x} - \mathbf{x}'\|_2^2\right)^{-\alpha}. \tag{2.6}$$

For the limit $\alpha \to \infty$ of its shape hyperparameter, this is the squared exponential covariance function [30].

Another interesting example is the exponential covariance function

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{\ell} \|\mathbf{x} - \mathbf{x}'\|_2\right), \tag{2.7}$$

which is very rough. It is a special case of the so-called Matérn class of covariance functions [30]. For $D = 1$, it is the covariance function of the Ornstein–Uhlenbeck process as a model for the velocity of a particle in Brownian motion [34].

To express a process that repeats itself in periods of T, one may consider the periodic covariance function [24]

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{2}{\ell^2} \sin^2\left(\frac{\pi}{T} \|\mathbf{x} - \mathbf{x}'\|_2\right)\right). \tag{2.8}$$

Gaussian processes are connected to linear regression. Consider Bayesian linear regression with a simple model $\mathbf{y} = \mathbf{X}^T\boldsymbol{\alpha} + \boldsymbol{\epsilon}$ for independent and identically distributed noise $\epsilon_n \sim \mathcal{N}(0, \sigma_n^2)$. When putting the prior $\boldsymbol{\alpha} \sim \mathcal{N}(\mathbf{0}, \sigma_0^2\mathbf{I})$ on the parameter vector, we infer $\mathbf{X}^T\boldsymbol{\alpha} \sim \mathcal{N}(\mathbf{0}, \sigma_0^2\mathbf{X}^T\mathbf{X})$ by the rule for linear transformations of a Gaussian random variable [33, Theorem 3.3.3]. Introducing $\mathbf{f} = \mathbf{X}^T\boldsymbol{\alpha}$, we conclude that this is just a Gaussian process with the zero mean function and the covariance function $k(\mathbf{x}, \mathbf{x}') = \sigma_0^2\mathbf{x}^T\mathbf{x}'$. Furthermore, Gaussian processes with a certain covariance function are connected to neural networks [26].

2.2. Multivariate Gaussian Distribution

Gaussian processes are convenient in the sense that many of the involved calculations can be done analytically instead of resorting to numerical approximations. It is thus useful know a couple of rules for calculating with multivariate Gaussian distributions, of which we first give a possible definition.

Definition 1. A D-dimensional random variable $\mathbf{x}$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ has a *multivariate Gaussian distribution* if and only if for all $\mathbf{r} \in \mathbb{R}^D$, the distribution of $\mathbf{r}^T\mathbf{x}$ is a univariate Gaussian $\mathcal{N}(\mathbf{r}^T\boldsymbol{\mu}, \mathbf{r}^T\boldsymbol{\Sigma}\mathbf{r})$ [33, Definition 3.2.5].

Note that this definition allows the covariance matrix $\mathbf{\Sigma}$ to be singular, in which case the distribution is called degenerate and does not have a probability density function. Otherwise, if $\mathbf{\Sigma}$ is invertible, the density is given as follows.

Proposition 1. *The density of a multivariate Gaussian distribution with mean $\mu \in \mathbb{R}^D$ and symmetric positive-definite covariance $\mathbf{\Sigma} \in \mathbb{R}^{D \times D}$ is*

$$\mathcal{N}(x \mid \mu, \mathbf{\Sigma}) = (2\pi)^{-D/2} |\mathbf{\Sigma}|^{-1/2} \exp\left(-\frac{1}{2}(x-\mu)^\mathsf{T} \mathbf{\Sigma}^{-1}(x-\mu)\right)$$

[33, Theorem 3.2.4]. The logarithm of the density is

$$\log \mathcal{N}(x \mid \mu, \mathbf{\Sigma}) = -\frac{1}{2}\left((x-\mu)^\mathsf{T} \mathbf{\Sigma}^{-1}(x-\mu) + \log|\mathbf{\Sigma}| + D\log(2\pi)\right).$$

Based on the density, we deduce a couple of corollaries, which shall be used without reference later.

Proposition 2. *A Gaussian density in general satisfies*

1. $\mathcal{N}(x \mid \mu, \mathbf{\Sigma}) = \mathcal{N}(x-\mu \mid 0, \mathbf{\Sigma})$,
2. $\mathcal{N}(-\mu \mid 0, \mathbf{\Sigma}) = \mathcal{N}(\mu \mid 0, \mathbf{\Sigma})$, *and*
3. $\mathcal{N}(x \mid \mu, \mathbf{\Sigma}) = \mathcal{N}(\mu \mid 0, \mathbf{\Sigma}) \exp\left(x^\mathsf{T}\mathbf{\Sigma}^{-1}\left(\mu - \frac{1}{2}x\right)\right)$.

Proof. These identities are immediate consequences of Proposition 1. □

If a conditional distribution is a Gaussian with its mean itself being Gaussian distributed, then the marginal distribution is a Gaussian.

Proposition 3. *If $\mathbf{t} \sim \mathcal{N}(\mu, \mathbf{\Sigma})$ and $\mathbf{u} \mid \mathbf{t} \sim \mathcal{N}(\mathbf{t}, \mathbf{V})$, then $\mathbf{u} \sim \mathcal{N}(\mu, \mathbf{\Sigma} + \mathbf{V})$ [5, Equation (2.115)].*

Conditioning one random variable on another one with joint Gaussian distribution gives yet another Gaussian as follows.

Proposition 4. *If*

$$\begin{bmatrix} \mathbf{t} \\ \mathbf{u} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu \\ \mathbf{r} \end{bmatrix}, \begin{bmatrix} \mathbf{\Sigma} & \mathbf{A} \\ \mathbf{A}^\mathsf{T} & \mathbf{V} \end{bmatrix}\right)$$

then

$$\mathbf{t} \mid \mathbf{u} \sim \mathcal{N}\left(\mu + \mathbf{A}\mathbf{V}^{-1}(\mathbf{u} - \mathbf{r}), \mathbf{\Sigma} - \mathbf{A}\mathbf{V}^{-1}\mathbf{A}^\mathsf{T}\right)$$

[33, Theorem 3.3.4].

The next result helps us with a so-called Gaussian integral. The expression $|\mathbf{\Lambda}|\, \mathcal{N}(\mu \mid 0, \mathbf{\Lambda})$ shall frequently reappear later.

Proposition 5. *If $\mathbf{\Lambda}$ is symmetric positive-definite, then*

$$\int_{\mathbb{R}^D} \exp\left(\mathbf{x}^\top \left(\boldsymbol{\mu} - \frac{1}{2}\mathbf{\Lambda}\mathbf{x}\right)\right) \mathrm{d}^D\mathbf{x} = \frac{1}{|\mathbf{\Lambda}|\,\mathcal{N}(\boldsymbol{\mu} \mid \mathbf{0}, \mathbf{\Lambda})}$$

[36, page 14].

As with the entropy of a discrete random variable, the differential entropy of a continuous random variable is related to its shortest description length [11]. The unit of information we use here is the nat, which measures to the base of Euler's number e. In case of a Gaussian distribution, the differential entropy does not depend on the mean, but only on the determinant of the covariance.

Proposition 6. *The differential entropy of a Gaussian distribution in D dimensions is*

$$h(\mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})) = \frac{1}{2}(\log|\mathbf{\Sigma}| + D\log(2\pi e)),$$

where the unit is nats [11, Equation (8.44)].

2.3. Symmetric Positive-Definite Matrices

The covariance matrix of a Gaussian is always symmetric positive-semidefinite, which can be seen as a generalization of nonnegative real numbers to matrices. If additionally, the covariance matrix is invertible, then it is symmetric positive-definite.

Definition 2. A matrix $\mathbf{\Sigma} \in \mathbb{R}^{D\times D}$ is called *symmetric positive-definite* (or *symmetric positive-semidefinite,* respectively) if and only if it is symmetric and $\mathbf{x}^\top\mathbf{\Sigma}\mathbf{x} > 0$ (or $\mathbf{x}^\top\mathbf{\Sigma}\mathbf{x} \geqslant 0$, respectively) for all $\mathbf{x} \neq \mathbf{0}$.

Symmetric positive-definite matrices are closed under inversion, multiplication by a scalar, and addition to a symmetric positive-semidefinite matrix. For instance, the covariance matrix $\mathbf{K}_y = \mathbf{K}_f + \sigma_n^2\mathbf{I}$ for Gaussian processes is symmetric positive-definite. Furthermore, the diagonal entries of symmetric positive-definite matrices are positive. However, a product of symmetric positive-definite matrices may generally not be symmetric positive-definite, so that they do not form a group under matrix multiplication.

Proposition 7. *If $\mathbf{\Sigma}$ is symmetric positive-definite, then*

1. *$\mathbf{\Sigma}$ is invertible and $\mathbf{\Sigma}^{-1}$ is symmetric positive-definite [19, page 430],*
2. *$\gamma\mathbf{\Sigma}$ and $\mathbf{\Sigma} + \mathbf{V}$ are symmetric positive-definite for all $\gamma > 0$ and for all symmetric positive-semidefinite $\mathbf{V}$ [19, Observation 7.1.3], and*
3. *every submatrix of $\mathbf{\Sigma}$ obtained by removing rows and columns corresponding to the same index set is symmetric positive-definite [19, Observation 7.1.2].*

The next little helper is about combining a symmetric positive-definite matrix with a possibly rectangular matrix.

Proposition 8. *If $\mathbf{\Sigma}$ is symmetric positive-definite and $\mathbf{A}$ has full row rank, then $\mathbf{A\Sigma A}^\intercal$ is symmetric positive-definite [19, Observation 7.1.8.(b)].*

Every symmetric positive-definite matrix $\mathbf{\Sigma} \in \mathbb{R}^{D\times D}$ has a unique decomposition of the form $\mathbf{\Sigma} = \mathbf{L}\mathbf{L}^\intercal$ called Cholesky decomposition, such that $\mathbf{L} \in \mathbb{R}^{D\times D}$ is a lower triangular matrix with positive diagonal entries [19, Corollary 7.2.9]. The Cholesky decomposition of $\mathbf{\Sigma}$ can be computed in $D^3/2$ floating-point operations, which is half of the complexity for the LU decomposition [18, Algorithm 10.2].

Once the Cholesky decomposition of $\mathbf{\Sigma}$ is available, we can numerically evaluate a Gaussian density with this covariance $\mathbf{\Sigma}$. A system of linear equations $\mathbf{\Sigma x} = \mathbf{t}$ for $\mathbf{x}$ can be solved in $\Theta\left(D^2\right)$ floating-point operations: first solve $\mathbf{Lu} = \mathbf{t}$ for $\mathbf{u}$ and then $\mathbf{L}^\intercal\mathbf{x} = \mathbf{u}$ for $\mathbf{x}$, where both times a system with a triangular matrix is easy to solve. Computing the determinant of $\mathbf{L}$, on the other hand, requires $\Theta(D)$ floating-point operations.

Proposition 9. *If $\mathbf{\Sigma} \in \mathbb{R}^{D\times D}$ is a symmetric positive-definite matrix with Cholesky decomposition $\mathbf{\Sigma} = \mathbf{L}\mathbf{L}^\intercal$, then its determinant equals*

$$|\mathbf{\Sigma}| = \prod_{d=1}^{D} l_{d,d}^2 .$$

Proof. In general, the determinant of a product of matrices is the product of their determinants [19, page 11]. Therefore, we have $|\mathbf{\Sigma}| = |\mathbf{L}|\,|\mathbf{L}^\intercal|$. As the determinant of a triangular matrix is the product of its diagonal entries [19, page 31], the statement follows. □

The Cholesky decomposition enjoys a fantastic numerical stability. That is, there exists an exceptionally good guarantee how well $\widetilde{\mathbf{L}}\widetilde{\mathbf{L}}^\intercal$ approximates $\mathbf{\Sigma}$ for the numerically computed $\widetilde{\mathbf{L}}$ [18, Theorem 10.3]. The LAPACK standard software library provides efficient implementations of algorithms centered around numerical linear algebra [1]. Its routine `DPOTRF` performs a Cholesky decomposition of a symmetric positive-definite matrix, based on which `DPOTRS` solves linear systems of equations and `DPOTRI` computes the inverse. Available since LAPACK version 3.2, the routines `DPSTF2` and `DPSTRF` do a so-called pivoted Cholesky decomposition for the more general case of symmetric positive-semidefinite matrices [23].

2.4. Model Selection for Gaussian Processes

For Gaussian process regression, the problem of model selection is to determine a mean function, a covariance function, and a noise level σ_n. Determining a mean and a covariance function means to select both their functional form and their hyperparameters, if any. For instance, one may select as a functional form a zero mean function and a squared exponential covariance function of Equation (2.5) on page 12. The hyperparameter optimization is then to select ℓ, σ_f, and σ_n. Solving the model selection should not only improve the predictions of the model, but also let us interpret the data by discovering patterns such as periodicity [30].

2.4.1. General Aspects

The functional form selection may be done manually, in particular if one is an expert in the domain. More recently, search systems have been developed to automatically explore a large number of models and combinations thereof [12]. Besides the subjective examination by human users based on feedback such as visualizations, it is crucial to let a system assign an objective cost to a model given the data – the lower the cost, the better.

Cost functions occur at different levels of model selection. First of all, a cost function serves as an objective function to compare hyperparameters for a single functional form. On one level higher, a cost function compares functional forms on their optimized hyperparameters. To make this distinction clearer, let the data set be $\mathcal{D}$. We would like to decide for one of a discrete set $\mathcal{H}$ of possible functional forms and optimize its hyperparameters. Fixing the hyperparameters of a functional form $h \in \mathcal{H}$ to θ produces a Gaussian process denoted by $h(\theta)$. For each functional form $h \in \mathcal{H}$, we find an optimal hyperparameter vector

$$\theta_h^\star \in \arg\min_\theta R_2\left(h(\theta), \mathcal{D}\right) \tag{2.9}$$

according to the cost function R_2. Once the hyperparameters are optimized, we take them to choose a functional form by

$$h^\star \in \arg\min_{h \in \mathcal{H}} R_3\left(h(\theta_h^\star), \mathcal{D}\right) \tag{2.10}$$

with the cost function R_3. From the perspective of model selection by multi-level inference [15], these two levels can be seen as instances of the second and third level of inference, respectively, hence the names R_2 and R_3. The cost functions R_2 and R_3 can be the same, but do not need to be. We use the name R for a general cost function at any level of inference.

To minimize these cost functions, numerical optimization algorithms can be applied. In the family of quasi-Newton methods, the algorithm of limited-memory BFGS solves unconstrained optimization problems. It memorizes information from the last m iterations, where m is defined by the user [27]. In D dimensions, it does $\Theta(mD)$ work per iteration, oftentimes with a faster overall performance than the BFGS algorithm on which it is based [32]. Limited-memory BFGS needs to evaluate the value and gradient of the objective function, while it approximates the inverse of the Hessian matrix itself.

2.4.2. Maximum Evidence and Cross-Validation

A well-known criterion for model selection is maximum evidence, also known as marginal likelihood. Recall that we already have the evidence from Equation (2.3) on page 12. The criterion of maximum evidence is then to select a model that minimizes the negative log-evidence

$$R_{ME} = -\log \mathcal{N}(\mathbf{y} \mid \mathbf{m}, \mathbf{K}_y). \tag{2.11}$$

Note that the mean $\mathbf{m}$ and the covariance $\mathbf{K}_y$ in general depend on the hyperparameters, even if the notation hides this fact for simplicity.

The criterion of cross-validation estimates the generalization error to minimize it. K-fold cross-validation partitions the data into K subsets of approximately the same cardinality. For every subset, a model is trained on the other $K-1$ subsets together in order to evaluate a loss function on the current subset. Finally, the resulting K losses are averaged, for example by their mean. One could even do more than one round of K-fold cross-validation on multiple random partitions. In the extreme case of $K = N$ folds, K-fold cross-validation uses a single case per validation, which is called leave-one-out (LOO) cross-validation. As a loss function for cross-validation, the negative log-probability seems natural, but any other cost function such as the mean squared error is possible as well [30]. The decision depends on the application.

For the kth fold, let $\mathbf{X}_k$ and $\mathbf{y}_k$ belong to the kth subset of the data, while the data without this subset is made of $\mathbf{X}_{-k}$ and $\mathbf{y}_{-k}$. Our default is to do K-fold cross-validation for the negative log-probability loss with the objective to minimize

$$R_{CV} = -\frac{1}{K}\sum_{k=1}^{K} \log p(\mathbf{y}_k \mid \mathbf{X}, \mathbf{y}_{-k}). \tag{2.12}$$

To calculate it, we first reuse the posterior of Equation (2.4) on page 12 to get the density of $\mathbf{f}_k \mid \mathbf{X}, \mathbf{y}_{-k}$. We simply need to add the noise $\sigma_n^2 \mathbf{I}$ to its covariance matrix according to Proposition 3 on page 15, so that we have

$$\mathbf{y}_k \mid \mathbf{X}, \mathbf{y}_{-k} \sim \mathcal{N}\left(\mathbf{m}_k + \mathbf{A}_k \mathbf{K}_{-k}^{-1}(\mathbf{y}_{-k} - \mathbf{m}_{-k}), \mathbf{K}_k - \mathbf{A}_k \mathbf{K}_{-k}^{-1} \mathbf{A}_k^T\right).$$

where

$$\begin{aligned} &\mathbf{m}_k = \mathbf{m}(\mathbf{X}_k), && \mathbf{m}_{-k} = \mathbf{m}(\mathbf{X}_{-k}), \\ &\mathbf{K}_k = \mathbf{K}_y(\mathbf{X}_k, \mathbf{X}_k), && \mathbf{K}_{-k} = \mathbf{K}_y(\mathbf{X}_{-k}, \mathbf{X}_{-k}), && \mathbf{A}_k = \mathbf{K}_f(\mathbf{X}_k, \mathbf{X}_{-k}). \end{aligned}$$

The last criterion of probably approximately correct (PAC) learning tries to bound the generalization error. One can do model selection by minimizing these generalization bounds. However, the risk is that "this procedure may not be well-justified if the generalization bounds are loose." [30, page 163] Furthermore, while the involved PAC-Bayesian theorem holds for Gaussian process classification, it seems unclear whether it can be applied to Gaussian process regression [31]. We shall thus not further consider PAC learning here.

How do the criteria of maximum evidence and cross-validation compare? Leave-one-out cross-validation has the same asymptotic time complexity as maximum evidence, namely $\Theta\left(N^3\right)$ due to the matrix decomposition of $\mathbf{K}_y$. For K-fold cross-validation, the trade-off between bias and variance generally depends on K. Leave-one-out cross-validation is normally less biased, but has a higher variance compared to when the number of folds K is reduced [21]. Maximum evidence puts more emphasis on the model than the data, so that it could be preferred over leave-one-out cross-validation if one has a lot of trust in the prior [9]. On the other hand, cross-validation procedures have been argued to perform better in cases of model misspecification [2].

2.5. Approximation Set Coding

Approximation set coding (ASC) is an abstract theory to do model selection. It selects a model that neither underfits nor overfits the given data. In other words, an optimal model should both be informative and have a reproducible inference. Clearly, this trade-off is at the basis of model selection itself and has been popularized by the principle of Occam's razor. What distinguishes approximation set coding is the idea to map the problem of model selection to an imaginary communication scenario, where the trade-off is between a high information rate and a low communication error. Independently of Gaussian processes, we review approximation set coding in the following, as developed by Buhmann [7].

2.5.1. Inferring Posterior Distributions

Let the data set be $\mathcal{D}$. We have a countable set of possible hypotheses $\mathcal{C}$, each of which is an interpretation of the data. A cost function $R\left(\mathbf{c}, \mathcal{D}\right)$ measures how plausible the hypothesis $\mathbf{c} \in \mathcal{C}$ is in light of the data $\mathcal{D}$. Now instead of just one cost function, we have (possibly infinitely) many cost functions $R_\theta\left(\mathbf{c}, \mathcal{D}\right)$ indexed by θ.

To make things clearer, an instance of model selection is to determine the number of clusters K in K-means clustering [10]. In this case, the data $\mathcal{D}$ is made of N points to be clustered in D dimensions. The hypothesis space $\mathcal{C}$ is the set of all possible assignments of the points to K clusters, so that it has cardinality K^N. Each number of clusters $K \in \mathbb{N}$ induces a cost function $R_K\left(\mathbf{c}, \mathcal{D}\right)$ defined as the sum of squared distances from each data point to its assigned cluster mean. We assume here that for each number of clusters K, the K cluster means are already estimated, for example by Lloyd's algorithm [22]. Approximation set coding provides us with a principle to select an optimal cost function, thereby selecting the number of clusters K.

Continuing with the general theory, once the cost functions are defined, we can find an empirical minimizer $\mathbf{c}^\perp \in \arg\min_\mathbf{c} R\left(\mathbf{c}, \mathcal{D}\right)$ per cost function. The empirical minimizer $\mathbf{c}^\perp$ depends on the noise that comes with the data $\mathcal{D}$, so that identifying $\mathbf{c}^\perp$ as the only interpretation of the data would not reflect the uncertainty of the data. What we would like to infer instead is a distribution over the hypothesis space such that hypotheses with less costs are assigned more probability mass. This is expressed as a posterior distribution $\Pr_\beta(\mathbf{c} \mid \mathcal{D})$ that depends on the so-called approximation precision $\beta > 0$. The role of β is to control the trade-off between informativeness and robustness of the inferred posterior $\Pr_\beta(\mathbf{c} \mid \mathcal{D})$: a high β assigns most of the probability mass to the hypotheses of low cost (informative), while reducing β results in spreading the probability mass more equally over the hypotheses (robust).

How do we choose the approximation precision β? To answer this, approximation set coding defines an imaginary communication scenario with a coding concept (the actual approximation set coding). It is based on transformations of the data $\mathcal{D}$ that correspond to transformations in the hypothesis space $\mathcal{C}$. Without explaining the scenario here, it essentially amounts to a bound on the communication error that depends on an expression I_β^θ called mutual information. Note that this mutual information is not equal to any mutual information between random variables, but rather just recycles the same name due to its resemblance to the mutual information in the context of channel coding. The idea is then to choose β such that the mutual information is maximal, that is,

$$\beta_\theta^\star \in \operatorname*{arg\,max}_{\beta>0} I_\beta^\theta. \tag{2.13}$$

The maximum $I_{\beta^\star}^\theta$ of the mutual information with respect to β is known as the approximation capacity.

2.5.2. Model Selection Principle

To do model selection, approximation set coding then simply tells us to select a cost function $R_\theta(\mathbf{c}, \mathcal{D})$ with maximal approximation capacity as in

$$\theta^\star \in \operatorname*{arg\,max}_{\theta} I_{\beta^\star}^\theta. \tag{2.14}$$

Since models normally come with cost functions such as a likelihood, this principle allows us to compare models.

The mutual information is given by

$$I_\beta^\theta = h(\tau) + \frac{2}{N} \log \eta_\beta. \tag{2.15}$$

While $h(\tau)$ is independent of β [14], both $h(\tau)$ and η_β generally depend on the cost function $R_\theta(\mathbf{c}, \mathcal{D})$. The expression $h(\tau)$ measures the complexity of the transformations in the communication scenario. For example, in the case of K-means clustering, $h(\tau)$ can be estimated by the entropy of the empirical minimizer $\mathbf{c}^\perp$ [10, Equation (4)].

To define the other expression η_β, we first need to state a strong assumption that approximation set coding makes. It is assumed that the data $\mathcal{D}$ with N cases can be partitioned into two subsets $\mathcal{D}_1$ and $\mathcal{D}_2$ of equal size $N/2$ such that the nth case of $\mathcal{D}_1$ corresponds to the nth case of $\mathcal{D}_2$. Expressing it in the language of approximation set coding, the nth cases of $\mathcal{D}_1$ and $\mathcal{D}_2$ are two measurements of the same object. In K-means clustering, for instance, the nth points of $\mathcal{D}_1$ and $\mathcal{D}_2$ are assumed to be generated from the same cluster. In that case, however, the assumption can be established by clustering the data $\mathcal{D}$ in order to construct $\mathcal{D}_1$ and $\mathcal{D}_2$ by randomly partitioning the points assigned to the same cluster into equally sized subsets.

Each of the subsets $\mathcal{D}_1$ and $\mathcal{D}_2$ gives rise to its own posterior distribution $\Pr_\beta(\mathbf{c} \mid \mathcal{D}_l)$ for $l = 1, 2$. This leads us to introducing

$$\eta_\beta = \sum_{\mathbf{c}\in\mathcal{C}} \Pr_\beta(\mathbf{c} \mid \mathcal{D}_1) \Pr_\beta(\mathbf{c} \mid \mathcal{D}_2)\,. \tag{2.16}$$

As the expression η_β measures the overlap of the two posteriors, it is also called the posterior agreement.[1]

A crucial ingredient is how to construct a posterior distribution for a given β. The standard approach so far is to come up with nonnegative weights $w_\beta(\mathbf{c}, \mathcal{D})$ such that hypotheses with less costs $R_\theta(\mathbf{c}, \mathcal{D})$ get more weight. For example, one may decide for so-called Boltzmann weights

$$w_\beta(\mathbf{c}, \mathcal{D}) = \exp(-\beta\, R_\theta(\mathbf{c}, \mathcal{D}))\,.$$

Once the weights are defined, we only need to normalize them for a probability distribution over the hypothesis space as in

$$\Pr_\beta(\mathbf{c} \mid \mathcal{D}) = \frac{w_\beta(\mathbf{c}, \mathcal{D})}{\sum_{\mathbf{c}\in\mathcal{C}} w_\beta(\mathbf{c}, \mathcal{D})}.$$

As we shall see later, models with a prior provide an alternative posterior via Bayes' theorem.

2.5.3. Uncountable Hypothesis Spaces

The original formulation of approximation set coding addresses a countable hypothesis space $\mathcal{C}$. Frank and Buhmann [14] solve the problem of selecting the rank K for a truncated singular value decomposition (SVD) by approximation set coding. The theory is thereby transferred to the uncountable hypothesis space of that particular application. Two interesting variants are presented. The first variant is to discretize the hypothesis space by an equispaced grid in a finite hypercube. However, it is unclear how to choose the limits of this hypercube for a model at hand.

The second, presumably more attractive variant for uncountable hypothesis spaces is to replace the sums over hypotheses by integrals with respect to a measure $\gamma F(\mathbf{c})$. $F(\mathbf{c})$ denotes a cumulative distribution function over the hypothesis space, scaled by a positive scalar γ. Frank and Buhmann [14] choose a Gaussian $F(\mathbf{c})$ for the integration of

[1]The Bhattacharyya coefficient between two distributions happens to be of a related form, but it additionally takes the square root of the probabilities [4].

both the weights and the expression η_β. More concretely, instead of directly transferring the weight sums to $\int_{\mathcal{C}} w_\beta\left(\mathbf{c}, \mathcal{D}\right)\, d^D\mathbf{c}$, they become

$$\int_{\mathcal{C}} w_\beta\left(\mathbf{c}, \mathcal{D}\right)\, d^D\left(\gamma F\left(\mathbf{c}\right)\right) = \gamma \int_{\mathcal{C}} w_\beta\left(\mathbf{c}, \mathcal{D}\right) p\left(\mathbf{c}\right)\, d^D\mathbf{c},$$

where $p\left(\mathbf{c}\right)$ is the probability density function of $F\left(\mathbf{c}\right)$. The expression η_β is transferred in the same way. In this case, the involved expression

$$\frac{w_\beta\left(\mathbf{c}, \mathcal{D}\right)}{\gamma \int_{\mathcal{C}} w_\beta\left(\mathbf{c}, \mathcal{D}\right)\, d^D F\left(\mathbf{c}\right)}$$

is not a density any more, since it does not integrate to 1 for the default integration measure. Therefore, η_β does not measure a posterior agreement in this case. A further contribution is to take the complexity of the transformations $h\left(\tau\right)$ as

$$h\left(\tau\right) = \frac{2}{N}\log\left(\gamma \int_{\mathcal{C}} d^D F\left(\mathbf{c}\right)\right) = \frac{2}{N}\log\left(\gamma \int_{\mathcal{C}} p\left(\mathbf{c}\right)\, d^D\mathbf{c}\right) = \frac{2}{N}\log\gamma.$$

In general, we assume that $F\left(\mathbf{c}\right)$ has a probability density function $p\left(\mathbf{c}\right)$ exactly over the hypothesis space.

3. Model Selection by Approximation Set Coding with a Prior

Given a model with parameters, we would like to solve the problem of selecting its hyperparameters by approximation set coding. The central requirement is that we have a prior over the parameters, as is the case for Bayesian linear regression. The treatment in this chapter is of an abstract nature, preparing the ground for Gaussian process models in Chapter 4 on page 30.

3.1. Using a Prior on the Hypothesis Space

Let a model have parameters $\boldsymbol{\alpha} \in \mathbb{R}^D$ and hyperparameters θ. We assume that it comes with a likelihood $p(\mathcal{D} \mid \boldsymbol{\alpha})$ and a prior $p(\boldsymbol{\alpha})$, both of which may depend on the hyperparameters. The goal is to select the hyperparameters based on the given data $\mathcal{D}$.

In the context of approximation set coding, we designate the parameters $\boldsymbol{\alpha}$ as the hypotheses. A natural cost function is the negative log-likelihood

$$R(\boldsymbol{\alpha}, \mathcal{D}) = -\log p(\mathcal{D} \mid \boldsymbol{\alpha}). \tag{3.1}$$

Since the hyperparameters θ affect the costs, we can optimize the hyperparameters by comparing cost functions. Table 3.1 relates this instantiation of approximation set coding to other examples of model selection.

Based on a fixed cost function, we define a family of posteriors $p_\beta(\boldsymbol{\alpha} \mid \mathcal{D})$ parametrized by the approximation precision β. There exist several ways to define the posteriors, one of which is via Boltzmann weights. For now, we abstract from this choice.

The data $\mathcal{D}$ of size N is partitioned into L subsets $\mathcal{D}_l$, each of its own cardinality N_l for $l = 1, \dots, L$. Unlike in the original theory of approximation set coding [7], we do not assume a direct correspondence of the cases in these subsets, but rather think of the

Application	Model θ to select	Hypothesis $\mathbf{c}$
K-means clustering [10]	Number K of clusters	Cluster assignment
Truncated SVD [14]	Rank K	Matrix decomposition
Model with parameter prior	Hyperparameters θ	Parameters $\boldsymbol{\alpha}$

Table 3.1.: Instantiations of approximation set coding for various applications.

subsets being indirectly connected through the parameters $\boldsymbol{\alpha}$. We look at the subsets as measurements of the parameters. This means that we are free to choose the subsets, including their cardinalities. For abstraction, the number L of subsets is arbitrary as well, which at the same time shortens the notation without adding much complexity to the derivations.

Given such a partition of the data $\mathcal{D}$, the posterior agreement in resemblance with Equation (2.16) on page 22 is

$$\eta_\beta = \int_{\mathbb{R}^D} \prod_{l=1}^{L} p_\beta \left(\boldsymbol{\alpha} \mid \mathcal{D}_l \right) \, d^D F \left(\boldsymbol{\alpha} \right), \tag{3.2}$$

where $F(\boldsymbol{\alpha})$ is the cumulative distribution function of the prior on $\boldsymbol{\alpha}$. If $F(\boldsymbol{\alpha})$ has a probability density function $p(\boldsymbol{\alpha})$, then

$$\eta_\beta = \int_{\mathbb{R}^D} \left(\prod_{l=1}^{L} p_\beta \left(\boldsymbol{\alpha} \mid \mathcal{D}_l \right) \right) p\left(\boldsymbol{\alpha}\right) \, d^D \boldsymbol{\alpha}.$$

Inspired by Frank and Buhmann [14], we choose a density as an integration measure to reweight the hypothesis space. Intuitively, it is natural to let the prior $p(\boldsymbol{\alpha})$ take this role, since we care more about the agreement of the posteriors on the hypotheses that are a priori more plausible.

While defining the complexity of the transformations $h(\tau)$ is not entirely clear, it is believed to be a design decision to some degree. One can simply resort to

$$h(\tau) = 0. \tag{3.3}$$

That way, approximation set coding reduces to a pure posterior agreement, which can be a good base to build upon. As an alternative, it makes sense to take $h(\tau)$ as a measure for the complexity of the integration measure. Thus, the differential entropy of the prior would be a candidate, that is,

$$h(\tau) = h\left(p\left(\boldsymbol{\alpha}\right)\right). \tag{3.4}$$

This is related to the shortest description length of the parameters $\boldsymbol{\alpha}$ under the prior and hence to the richness of the transformations τ on the hypothesis space. For simplicity, we omit the factor $2/N$ for $h(\tau)$, as it appears for $\log \eta_\beta$ in Equation (2.15) on page 21, too. Therefore, the mutual information equals

$$I_\beta^\theta = h(\tau) + \log \eta_\beta. \tag{3.5}$$

For model selection, we maximize the approximation capacity as in

$$\theta^\star \in \arg\max_\theta \left(\max_{\beta > 0} I_\beta^\theta \right). \tag{3.6}$$

Instead of a single partition of the data $\mathcal{D}$, one can average over the approximation capacity of J partitions. The abstract Algorithm 1 on the next page summarizes this model

Algorithm 1 Model selection by approximation set coding for the data $\mathcal{D}$. The approximation capacity is computed as the mean of J partitions into L subsets.

```
for $j \leftarrow 1, \dots, J$ do in parallel
    $\mathcal{D}_1^{(j)}, \dots, \mathcal{D}_L^{(j)} \leftarrow$ random partition of $\mathcal{D}$ into L subsets
end for

function $I(\theta)$
    for $j \leftarrow 1, \dots, J$ do in parallel
        $\gamma^{(j)} \leftarrow \max_{\beta>0} I_\beta^\theta \left(\mathcal{D}_1^{(j)}, \dots, \mathcal{D}_L^{(j)}\right)$        ▷ Equation (3.5) on page 25
    end for
    return $\frac{1}{J} \sum_{j=1}^{J} \gamma^{(j)}$
end function

$\theta^\star \leftarrow \arg\max_\theta I(\theta)$
return $\theta^\star$
```

selection. Because the computations for the J partitions are independent per round, they can easily be done in parallel.

Next, we shall look at several variants to construct posteriors $p_\beta(\boldsymbol{\alpha} \mid \mathcal{D})$ based on the cost function of Equation (3.1) on page 24.

3.2. Boltzmann Approximation Set Coding

One can use Boltzmann weights to transform the costs into weights as in

$$w_\beta(\boldsymbol{\alpha}, \mathcal{D}) = \exp(-\beta R(\boldsymbol{\alpha}, \mathcal{D})) = p(\mathcal{D} \mid \boldsymbol{\alpha})^\beta .$$

We then normalize the weights to obtain the posterior

$$p_\beta(\boldsymbol{\alpha} \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \boldsymbol{\alpha})^\beta}{\int_{\mathbb{R}^D} p(\mathcal{D} \mid \boldsymbol{\alpha})^\beta \, d^D\boldsymbol{\alpha}}. \tag{3.7}$$

Note that unlike Frank and Buhmann [14], we thereby get a density, because the integration is simply with respect to $\boldsymbol{\alpha}$ instead of the changed measure $F(\boldsymbol{\alpha})$.

We now look at such a posterior for the special case of a Gaussian likelihood. The resulting posterior is another Gaussian. Its mean is the empirical minimizer of the cost function – as expected, since the posterior is a monotonically increasing function of the likelihood. Its covariance matrix, on the other hand, changes with the approximation precision β.

Proposition 10. *Let* $w_\beta(\mathbf{c}) = \mathcal{N}(\mathbf{A}^\mathsf{T}\mathbf{c} \mid \mu, \boldsymbol{\Sigma})^\beta$ *for arbitrary* $\beta > 0$. *Under the assumption that* $\mathbf{A} \in \mathbb{R}^{D \times N}$ *has full row rank,*

$$\frac{w_\beta(\mathbf{c})}{\int_{\mathbb{R}^D} w_\beta(\mathbf{c}) \, d^D\mathbf{c}} = \mathcal{N}\left(\mathbf{c} \,\middle|\, \mathbf{r}, \frac{1}{\beta}\boldsymbol{\Lambda}^{-1}\right),$$

where $\mathbf{r} = \boldsymbol{\Lambda}^{-1}\mathbf{A}\boldsymbol{\Sigma}^{-1}\mu$ *and* $\boldsymbol{\Lambda} = \mathbf{A}\boldsymbol{\Sigma}^{-1}\mathbf{A}^\mathsf{T}$.

Proof. We first separate a factor from $w_\beta(\mathbf{c})$ that is independent of $\mathbf{c}$, namely

$$w_\beta(\mathbf{c}) = \mathcal{N}(\mathbf{A}^\mathsf{T}\mathbf{c} \mid \mu, \boldsymbol{\Sigma})^\beta = \mathcal{N}(\mu \mid 0, \boldsymbol{\Sigma})^\beta \exp\left(\beta\mathbf{c}^\mathsf{T}\left(\mathbf{A}\boldsymbol{\Sigma}^{-1}\mu - \frac{1}{2}\boldsymbol{\Lambda}\mathbf{c}\right)\right).$$

Then

$$\frac{w_\beta(\mathbf{c})}{\int_{\mathbb{R}^D} w_\beta(\mathbf{c})\ \mathrm{d}^D\mathbf{c}} = \frac{\exp\left(\beta\mathbf{c}^\mathsf{T}\left(\mathbf{A}\boldsymbol{\Sigma}^{-1}\mu - \frac{1}{2}\boldsymbol{\Lambda}\mathbf{c}\right)\right)}{\int_{\mathbb{R}^D}\exp\left(\beta\mathbf{c}^\mathsf{T}\left(\mathbf{A}\boldsymbol{\Sigma}^{-1}\mu - \frac{1}{2}\boldsymbol{\Lambda}\mathbf{c}\right)\right)\ \mathrm{d}^D\mathbf{c}}.$$

We now calculate the integral. It can be seen that $\beta\boldsymbol{\Lambda}$ is symmetric positive-definite by Proposition 7 on page 16 and Proposition 8 on page 17, since $\mathbf{A}$ is assumed to have full row rank. This allows us to apply Proposition 5 on page 16, so that

$$\int_{\mathbb{R}^D}\exp\left(\beta\mathbf{c}^\mathsf{T}\left(\mathbf{A}\boldsymbol{\Sigma}^{-1}\mu - \frac{1}{2}\boldsymbol{\Lambda}\mathbf{c}\right)\right)\ \mathrm{d}^D\mathbf{c} = \frac{1}{|\beta\boldsymbol{\Lambda}|\mathcal{N}(\beta\mathbf{A}\boldsymbol{\Sigma}^{-1}\mu \mid 0, \beta\boldsymbol{\Lambda})} = \frac{1}{\mathcal{N}\left(\mathbf{r}\ \middle|\ 0, \frac{1}{\beta}\boldsymbol{\Lambda}^{-1}\right)}.$$

Finally, we have

$$\begin{aligned}\frac{w_\beta(\mathbf{c})}{\int_{\mathbb{R}^D} w_\beta(\mathbf{c})\ \mathrm{d}^D\mathbf{c}} &= \mathcal{N}\left(\mathbf{r}\ \middle|\ 0, \frac{1}{\beta}\boldsymbol{\Lambda}^{-1}\right)\exp\left(\beta\mathbf{c}^\mathsf{T}\left(\mathbf{A}\boldsymbol{\Sigma}^{-1}\mu - \frac{1}{2}\boldsymbol{\Lambda}\mathbf{c}\right)\right)\\ &= \mathcal{N}\left(\mathbf{r}\ \middle|\ 0, \frac{1}{\beta}\boldsymbol{\Lambda}^{-1}\right)\exp\left(\beta\mathbf{c}^\mathsf{T}\boldsymbol{\Lambda}\left(\mathbf{r} - \frac{1}{2}\mathbf{c}\right)\right) = \mathcal{N}\left(\mathbf{c}\ \middle|\ \mathbf{r}, \frac{1}{\beta}\boldsymbol{\Lambda}^{-1}\right).\end{aligned}$$

□

We need one more intermediate proposition as a preparation to infer a specific posterior agreement. Integrating over a product of Gaussian densities is analytically feasible, which proves once more the convenience of Gaussians.

Proposition 11. *We have*

$$\int_{\mathbb{R}^D}\prod_{l=1}^{L}\mathcal{N}(\mathbf{x} \mid \mu_l, \boldsymbol{\Sigma}_l)\ \mathrm{d}^D\mathbf{x} = \frac{\prod_{l=1}^{L}\mathcal{N}(\mu_l \mid 0, \boldsymbol{\Sigma}_l)}{|\boldsymbol{\Lambda}|\mathcal{N}(\mathbf{r} \mid 0, \boldsymbol{\Lambda})},$$

where

$$\mathbf{r} = \sum_{l=1}^{L}\boldsymbol{\Sigma}_l^{-1}\mu_l, \qquad \boldsymbol{\Lambda} = \sum_{l=1}^{L}\boldsymbol{\Sigma}_l^{-1}.$$

Proof. We shorten $\gamma = \prod_{l=1}^{L}\mathcal{N}(\mu_l \mid 0, \boldsymbol{\Sigma}_l)$ and move this factor γ independent of $\mathbf{x}$ out of the integral as in

$$\begin{aligned}\int_{\mathbb{R}^D}\prod_{l=1}^{L}\mathcal{N}(\mathbf{x} \mid \mu_l, \boldsymbol{\Sigma}_l)\ \mathrm{d}^D\mathbf{x} &= \gamma\int_{\mathbb{R}^D}\prod_{l=1}^{L}\exp\left(\mathbf{x}^\mathsf{T}\boldsymbol{\Sigma}_l^{-1}\left(\mu_l - \frac{1}{2}\mathbf{x}\right)\right)\ \mathrm{d}^D\mathbf{x}\\ &= \gamma\int_{\mathbb{R}^D}\exp\left(\sum_{l=1}^{L}\mathbf{x}^\mathsf{T}\boldsymbol{\Sigma}_l^{-1}\left(\mu_l - \frac{1}{2}\mathbf{x}\right)\right)\ \mathrm{d}^D\mathbf{x}\\ &= \gamma\int_{\mathbb{R}^D}\exp\left(\mathbf{x}^\mathsf{T}\left(\mathbf{r} - \frac{1}{2}\boldsymbol{\Lambda}\mathbf{x}\right)\right)\ \mathrm{d}^D\mathbf{x}.\end{aligned}$$

The remaining integral can again be calculated by Proposition 5 on page 16 and the statement follows. □

Now we are ready to derive the case when both the likelihood and the prior are a Gaussian, which is to be applied to Gaussian processes later.

Proposition 12. *Let*

$$R(\mathbf{c}, l) = -\log \mathcal{N}\left(\mathbf{A}_l^\mathsf{T}\mathbf{c} \mid \mu_l, \boldsymbol{\Sigma}_l\right)$$

be a cost function with $\mathbf{A}_l \in \mathbb{R}^{D\times N_l}$ *having full row rank for* $l = 1, \ldots, L$*. Assume a Gaussian prior*

$$\mathbf{c} \sim \mathcal{N}(\mu_0, \boldsymbol{\Sigma}_0)$$

on the hypotheses. When using Boltzmann weights $w_\beta(\mathbf{c}, l) = \exp(-\beta R(\mathbf{c}, l))$*, the posterior agreement is*

$$\eta_\beta = \frac{\prod_{l=1}^L |\beta\boldsymbol{\Lambda}_l| \mathcal{N}(\beta\mathbf{r}_l \mid 0, \beta\boldsymbol{\Lambda}_l)}{|\mathbf{P}| \mathcal{N}(\mathbf{s} \mid 0, \mathbf{P})} \mathcal{N}(\mu_0 \mid 0, \boldsymbol{\Sigma}_0),$$

where

$$\mathbf{r}_l = \mathbf{A}_l\boldsymbol{\Sigma}_l^{-1}\mu_l, \qquad \boldsymbol{\Lambda}_l = \mathbf{A}_l\boldsymbol{\Sigma}_l^{-1}\mathbf{A}_l^\mathsf{T},$$

$$\mathbf{s} = \beta\sum_{l=1}^L \mathbf{r}_l + \boldsymbol{\Sigma}_0^{-1}\mu_0, \qquad \mathbf{P} = \beta\sum_{l=1}^L \boldsymbol{\Lambda}_l + \boldsymbol{\Sigma}_0^{-1}.$$

Proof. Using Proposition 10 on page 26, we can write the posterior agreement according to Equation (3.2) on page 25 as

$$\begin{aligned}\eta_\beta &= \int_{\mathbb{R}^D}\left(\prod_{l=1}^L \frac{w_\beta(\mathbf{c}, l)}{\int_{\mathbb{R}^D} w_\beta(\mathbf{c}, l)\, d^D\mathbf{c}}\right)\mathcal{N}(\mathbf{c} \mid \mu_0, \boldsymbol{\Sigma}_0)\, d^D\mathbf{c} \\ &= \int_{\mathbb{R}^D}\left(\prod_{l=1}^L \mathcal{N}\left(\mathbf{c} \,\middle|\, \boldsymbol{\Lambda}_l^{-1}\mathbf{r}_l, \frac{1}{\beta}\boldsymbol{\Lambda}_l^{-1}\right)\right)\mathcal{N}(\mathbf{c} \mid \mu_0, \boldsymbol{\Sigma}_0)\, d^D\mathbf{c}.\end{aligned}$$

Such an integral over a product of $L+1$ Gaussians equals

$$\eta_\beta = \frac{\prod_{l=1}^L \mathcal{N}\left(\boldsymbol{\Lambda}_l^{-1}\mathbf{r}_l \,\middle|\, 0, \frac{1}{\beta}\boldsymbol{\Lambda}_l^{-1}\right)}{|\mathbf{P}| \mathcal{N}(\mathbf{s} \mid 0, \mathbf{P})} \mathcal{N}(\mu_0 \mid 0, \boldsymbol{\Sigma}_0)$$

by Proposition 11 on page 27. Finally, we apply the fact that

$$\mathcal{N}\left(\boldsymbol{\Lambda}_l^{-1}\mathbf{r}_l \,\middle|\, 0, \frac{1}{\beta}\boldsymbol{\Lambda}_l^{-1}\right) = |\beta\boldsymbol{\Lambda}_l| \mathcal{N}(\beta\mathbf{r}_l \mid 0, \beta\boldsymbol{\Lambda}_l).$$

□

3.3. β-Noise Approximation Set Coding

There are models that intrinsically come with a notion of noise. If there is a hyperparameter σ_n that represents a noise level, then it may be worth trying to identify $\beta = 1/\sigma_n$. Remember that the approximation precision β intuitively tells us with how much resolution we can distinguish hypotheses with statistical significance. If the noise level is lower, then the resolution should be higher. Unlike in K-means clustering or truncated singular value decomposition, for example, modeling noise is omnipresent in domains such as linear regression.

Consequently, in case of a model with a noise level σ_n, it is justified to turn the cost function of Equation (3.1) on page 24 into a posterior distribution without the approximation precision β. Since this variant injects β via the noise instead, we call it β-noise approximation set coding. We define the weights with the help of the exponential function as

$$w_1(\boldsymbol{\alpha}, \mathcal{D}) = \exp(-R(\boldsymbol{\alpha}, \mathcal{D})) = p(\mathcal{D} \mid \boldsymbol{\alpha}).$$

This is nothing else than the expression for the likelihood. However, we look at it as a function of the parameters $\boldsymbol{\alpha}$ instead of the data $\mathcal{D}$ and thus need to normalize it to

$$p_1(\boldsymbol{\alpha} \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \boldsymbol{\alpha})}{\int_{\mathbb{R}^D} p(\mathcal{D} \mid \boldsymbol{\alpha})\, d^D\boldsymbol{\alpha}}. \tag{3.8}$$

Technically, β-noise approximation set coding is equivalent to fixing $\beta = 1$ in Boltzmann approximation set coding. To see this, compare the posterior $p_1(\boldsymbol{\alpha} \mid \mathcal{D})$ to the one of Equation (3.7) on page 26.

The special case of $L = 1$ subset reveals an interesting connection. The posterior agreement of Equation (3.2) on page 25 is then related to the evidence $p(\mathcal{D})$ as in

$$\eta_1 = \int_{\mathbb{R}^D} p_1(\boldsymbol{\alpha} \mid \mathcal{D})\, p(\boldsymbol{\alpha})\, d^D\boldsymbol{\alpha} = \frac{\int_{\mathbb{R}^D} p(\mathcal{D} \mid \boldsymbol{\alpha})\, p(\boldsymbol{\alpha})\, d^D\boldsymbol{\alpha}}{\int_{\mathbb{R}^D} p(\mathcal{D} \mid \boldsymbol{\alpha})\, d^D\boldsymbol{\alpha}} = \frac{p(\mathcal{D})}{\int_{\mathbb{R}^D} p(\mathcal{D} \mid \boldsymbol{\alpha})\, d^D\boldsymbol{\alpha}}.$$

In contrast to model selection by maximum evidence, this posterior additionally penalizes with the expression $\int_{\mathbb{R}^D} p(\mathcal{D} \mid \boldsymbol{\alpha})\, d^D\boldsymbol{\alpha}$ to select the hyperparameters $\boldsymbol{\theta}$.

3.4. Bayesian Approximation Set Coding

The last variant is to derive the posterior by making use of the prior. Applying Bayes' theorem, we easily find

$$p(\boldsymbol{\alpha} \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \boldsymbol{\alpha})\, p(\boldsymbol{\alpha})}{p(\mathcal{D})} = \frac{p(\mathcal{D} \mid \boldsymbol{\alpha})\, p(\boldsymbol{\alpha})}{\int_{\mathbb{R}^D} p(\mathcal{D} \mid \boldsymbol{\alpha})\, p(\boldsymbol{\alpha})\, d^D\boldsymbol{\alpha}}. \tag{3.9}$$

We refer to the corresponding criterion as Bayesian approximation set coding.

4. Approximation Set Coding for Gaussian Process Regression

At the core of Gaussian processes lies the prior of Equation (2.1) on page 11. Together with the likelihood for Gaussian process regression, we can apply the framework of Chapter 3 on page 24.

4.1. Abstract Algorithm for Model Selection

Suppose we are given the inputs $\mathbf{X} \in \mathbb{R}^{D \times N}$ with known outputs $\mathbf{y} \in \mathbb{R}^{N}$. In addition, let $\widetilde{\mathbf{X}} \in \mathbb{R}^{D \times M}$ be arbitrary inputs with latent function values $\widetilde{\mathbf{f}} \in \mathbb{R}^{M}$. Since the function values $\widetilde{\mathbf{f}}$ are the parameters, we identify them as the hypotheses in approximation set coding. We write $\widetilde{\mathbf{f}}$ instead of $\mathbf{f}$, as these function values $\widetilde{\mathbf{f}}$ do not need to correspond to the inputs $\mathbf{X}$. In fact, the choice of $\widetilde{\mathbf{X}}$ and the size M is a design decision with trade-offs as we shall see in Chapter 5 on page 39. In the context of approximation set coding, the columns of $\widetilde{\mathbf{X}}$ are our M objects. For a fixed M less than or equal to N, a simple way to choose the objects $\widetilde{\mathbf{X}}$ is by sampling M columns of $\mathbf{X}$ uniformly at random without replacement.

Instead of vectors $\widetilde{\mathbf{f}}$, one could consider taking functions $f : \mathbb{R}^{D} \to \mathbb{R}$ as hypotheses, but it is not clear how to do the integration over functions that is required for the posterior agreement.

The given data is partitioned into L subsets represented by $(\mathbf{X}_l, \mathbf{y}_l)$, each with its individual number N_l of cases. Although the sizes N_l can be chosen to approximately equal N/L, the framework allows them in principle to vary significantly. In summary, Algorithm 2 on the next page describes on an abstract level how hyperparameters can be optimized.

The next goal is to instantiate the variants identified in Chapter 3 on page 24. Recall from Equation (3.5) on page 25 that the criterion of approximation set coding is the mutual information

$$I_{\beta}^{\theta} = h(\tau) + \log \eta_{\beta}, \tag{4.1}$$

where there are several variants for the transformation complexity $h(\tau)$ and for the pos-

Algorithm 2 Hyperparameter optimization for Gaussian process regression by approximation set coding for the data $\mathcal{D} = \{(x_n, y_n) \mid n = 1, \dots, N\}$. The approximation capacity is computed as the mean of J partitions into L subsets for M objects.

```
for $j \leftarrow 1, \dots, J$ do in parallel
    $\mathcal{D}_1^{(j)}, \dots, \mathcal{D}_L^{(j)} \leftarrow$ random partition of $\mathcal{D}$ into L subsets of respective size $N_l$
    $\widetilde{\mathbf{X}}^{(j)} \leftarrow$ M distinct columns of $\mathbf{X}$ chosen at random (as a $D \times M$ matrix)
end for

function $I(\boldsymbol{\theta})$
    for $j \leftarrow 1, \dots, J$ do in parallel
        $\gamma^{(j)} \leftarrow \max_{\beta > 0} I_\beta^{\boldsymbol{\theta}} \left( \mathcal{D}_1^{(j)}, \dots, \mathcal{D}_L^{(j)}; \widetilde{\mathbf{X}}^{(j)} \right)$        ▷ Equation (4.1) on page 30
    end for
    return $\frac{1}{J} \sum_{j=1}^{J} \gamma^{(j)}$
end function

$\boldsymbol{\theta}^\star \leftarrow \arg\max_{\boldsymbol{\theta}} I(\boldsymbol{\theta})$
return $\boldsymbol{\theta}^\star$
```

terior agreement η_β. The following derivations share the definitions

$$\begin{aligned} \mathbf{m}_l &= \mathbf{m}(\mathbf{X}_l) \in \mathbb{R}^{N_l}, & \mathbf{K}_l &= \mathbf{K}_y(\mathbf{X}_l, \mathbf{X}_l) \in \mathbb{R}^{N_l \times N_l}, \\ \widetilde{\mathbf{m}} &= \mathbf{m}\left(\widetilde{\mathbf{X}}\right) \in \mathbb{R}^{M}, & \widetilde{\mathbf{K}} &= \mathbf{K}_f\left(\widetilde{\mathbf{X}}, \widetilde{\mathbf{X}}\right) \in \mathbb{R}^{M \times M}, \\ & & \widetilde{\mathbf{K}}_l &= \mathbf{K}_f\left(\mathbf{X}_l, \widetilde{\mathbf{X}}\right) \in \mathbb{R}^{N_l \times M}. \end{aligned}$$

To numerically optimize the resulting objective functions, we analytically calculate their gradients. More precisely, we provide the derivative with respect to an arbitrary hyperparameter θ, which is an entry of the hyperparameter vector $\boldsymbol{\theta}$. A little collection of helpful derivatives can be found in Appendix A on page 61.

4.2. Variants for the Transformation Complexity

For a pure posterior agreement, one may simply ignore the complexity of the transformations by taking $h(\tau) = 0$. As an alternative suggested by Equation (3.4) on page 25, $h(\tau)$ could summarize the integration measure via the differential entropy of the prior on $\widetilde{\mathbf{f}}$. Using Proposition 6 on page 16, a variant is therefore

$$h(\tau) = h\left(\mathcal{N}\left(\widetilde{\mathbf{m}}, \widetilde{\mathbf{K}}\right)\right) = \frac{1}{2}\left(\log\left|\widetilde{\mathbf{K}}\right| + M \log(2\pi e)\right). \tag{4.2}$$

The derivative of the latter is

$$\frac{\partial\, h(\tau)}{\partial \theta} = \frac{1}{2} \operatorname{tr}\left(\widetilde{\mathbf{K}}^{-1} \frac{\partial \widetilde{\mathbf{K}}}{\partial \theta}\right).$$

4.3. Variants for the Posterior Agreement

We continue to derive the variants for the posterior agreement η_β presented in Chapter 3 on page 24.

4.3.1. Boltzmann Approximation Set Coding

Boltzmann approximation set coding infers a posterior by normalizing Boltzmann weights associated with a cost function. A natural cost function for Gaussian process regression is the negative log-likelihood

$$R\left(\widetilde{f},(X_l,y_l)\right) = -\log p\left(y_l \,\middle|\, \widetilde{X},\widetilde{f},X_l\right).$$

In order to apply Proposition 12 on page 28, we would like to express the cost function in the required form of a Gaussian. Since we have a Gaussian process,

$$\begin{bmatrix} y_l \\ \widetilde{f} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m_l \\ \widetilde{m} \end{bmatrix}, \begin{bmatrix} K_l & \widetilde{K}_l \\ \widetilde{K}_l^T & \widetilde{K} \end{bmatrix}\right).$$

For this Gaussian distribution to be non-degenerate, we need $\widetilde{K}$ to be symmetric positive-definite. Thus, we constrain the choice of $\widetilde{X}$ such that $\widetilde{K}$ is invertible. Note that as $\widetilde{K}$ is a covariance matrix, it is at least symmetric positive-semidefinite by construction. Using Proposition 4 on page 15, we find the likelihood

$$y_l \,\middle|\, \widetilde{X},\widetilde{f},X_l \sim \mathcal{N}\left(m_l + \widetilde{K}_l\widetilde{K}^{-1}\left(\widetilde{f}-\widetilde{m}\right), K_l - \widetilde{K}_l\widetilde{K}^{-1}\widetilde{K}_l^T\right).$$

Introducing

$$\begin{aligned} A_l &= \widetilde{K}^{-1}\widetilde{K}_l^T \in \mathbb{R}^{M\times N_l}, \\ \mu_l &= y_l - m_l + A_l^T\widetilde{m} \in \mathbb{R}^{N_l}, \\ \Sigma_l &= K_l - \widetilde{K}_l A_l \in \mathbb{R}^{N_l\times N_l}, \end{aligned}$$

we can rewrite the cost function as

$$\begin{aligned} R\left(\widetilde{f},(X_l,y_l)\right) &= -\log\mathcal{N}\left(y_l \,\middle|\, m_l + A_l^T\left(\widetilde{f}-\widetilde{m}\right),\Sigma_l\right) \\ &= -\log\mathcal{N}\left(m_l + A_l^T\left(\widetilde{f}-\widetilde{m}\right) - y_l \,\middle|\, 0,\Sigma_l\right) \\ &= -\log\mathcal{N}\left(A_l^T\widetilde{f} \,\middle|\, \mu_l,\Sigma_l\right). \end{aligned}$$

Together with the Gaussian process prior

$$\widetilde{f} \sim \mathcal{N}\left(\widetilde{m},\widetilde{K}\right),$$

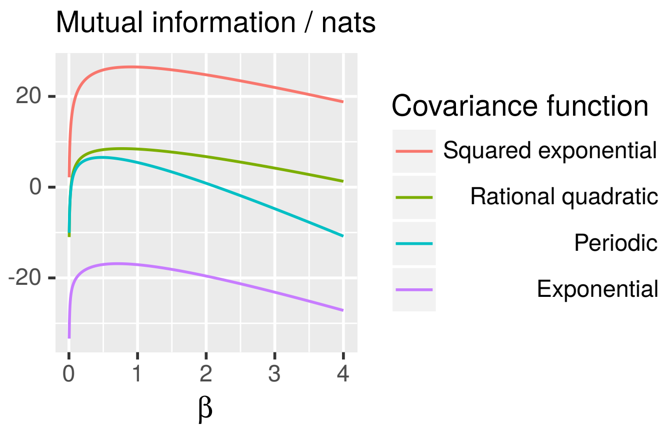


Figure 4.1.: Examples for I_β^θ as a function of β for various covariance functions based on synthetic data sets. In each case, $h(\tau) = 0$ is used. The objective functions appear to be well-behaved from the perspective of numerical optimization.

we are ready make use of Proposition 12 on page 28, assuming that $\mathbf{A}_l$ has full row rank for $l = 1, \ldots, L$. Hence, the posterior agreement equals

$$\eta_\beta = \frac{\prod_{l=1}^{L} |\beta \boldsymbol{\Lambda}_l| \mathcal{N}(\beta \mathbf{r}_l \mid 0, \beta \boldsymbol{\Lambda}_l)}{|\boldsymbol{\Lambda}| \mathcal{N}(\mathbf{r} \mid 0, \boldsymbol{\Lambda})} \mathcal{N}\left(\tilde{\mathbf{m}} \,\middle|\, 0, \tilde{\mathbf{K}}\right), \tag{4.3}$$

where

$$\mathbf{r}_l = \mathbf{A}_l \boldsymbol{\Sigma}_l^{-1} \mu_l, \qquad \boldsymbol{\Lambda}_l = \mathbf{A}_l \boldsymbol{\Sigma}_l^{-1} \mathbf{A}_l^\intercal,$$

$$\mathbf{r} = \beta \sum_{l=1}^{L} \mathbf{r}_l + \tilde{\mathbf{K}}^{-1} \tilde{\mathbf{m}}, \qquad \boldsymbol{\Lambda} = \beta \sum_{l=1}^{L} \boldsymbol{\Lambda}_l + \tilde{\mathbf{K}}^{-1}.$$

When does the premise hold that $\mathbf{A}_l$ has full row rank? Because $\mathbf{A}_l$ is an $M \times N_l$ matrix, it has full row rank by definition if and only if its row rank is M. At the same time, the row rank is at most N_l. That provides us with another constraint on the choice of $\tilde{\mathbf{X}}$: the number M of objects should not exceed any N_l.

Determining the Optimal Approximation Precision

For fixed hyperparameters θ, the optimal approximation precision is

$$\beta^\star \in \arg\max_{\beta > 0} I_\beta^\theta.$$

Figure 4.1 illustrates examples of the behavior of I_β^θ as a function of β.

As β is positive, we technically optimize with respect to $\log \beta$ instead of β in order to make algorithms for unconstrained optimization applicable. Since $h(\tau)$ never depends on β, the derivative of the mutual information is

$$\begin{aligned}\frac{\partial I_\beta^\theta}{\partial \log \beta} &= \frac{\partial h(\tau)}{\partial \log \beta} + \frac{\partial \log \eta_\beta}{\partial \log \beta} \\ &= \sum_{l=1}^{L} \frac{\partial \log\left(|\beta \boldsymbol{\Lambda}_l| \mathcal{N}(\beta \mathbf{r}_l \mid 0, \beta \boldsymbol{\Lambda}_l)\right)}{\partial \log \beta} - \frac{\partial \log\left(|\boldsymbol{\Lambda}| \mathcal{N}(\mathbf{r} \mid 0, \boldsymbol{\Lambda})\right)}{\partial \log \beta}.\end{aligned}$$

Using Proposition 16 on page 62, we get

$$\frac{\partial \log\left(|\beta \boldsymbol{\Lambda}_l| \mathcal{N}(\beta \mathbf{r}_l \mid 0, \beta \boldsymbol{\Lambda}_l)\right)}{\partial \log \beta} = \frac{\beta}{2}\left(\mathbf{r}_l^T \boldsymbol{\Lambda}_l^{-1}(\mathbf{r}_l - 2\mathbf{r}_l) + \frac{M}{\beta}\right) = \frac{1}{2}\left(M - \beta \mathbf{r}_l^T \boldsymbol{\Lambda}_l^{-1} \mathbf{r}_l\right)$$

and

$$\frac{\partial \log\left(|\boldsymbol{\Lambda}| \mathcal{N}(\mathbf{r} \mid 0, \boldsymbol{\Lambda})\right)}{\partial \log \beta} = \frac{\beta}{2}\left(\mathbf{r}^T \boldsymbol{\Lambda}^{-1}\left(\sum_{l=1}^{L} \boldsymbol{\Lambda}_l \boldsymbol{\Lambda}^{-1} \mathbf{r} - 2\sum_{l=1}^{L} \mathbf{r}_l\right) + \operatorname{tr}\left(\boldsymbol{\Lambda}^{-1} \sum_{l=1}^{L} \boldsymbol{\Lambda}_l\right)\right).$$

Gradient with Respect to the Hyperparameters

Given an arbitrary hyperparameter θ, the derivative of the logarithm of the posterior agreement is

$$\begin{aligned}\frac{\partial \log \eta_\beta}{\partial \theta} =& \sum_{l=1}^{L} \frac{\partial \log\left(|\beta \boldsymbol{\Lambda}_l| \mathcal{N}(\beta \mathbf{r}_l \mid 0, \beta \boldsymbol{\Lambda}_l)\right)}{\partial \theta} - \frac{\partial \log\left(|\boldsymbol{\Lambda}| \mathcal{N}(\mathbf{r} \mid 0, \boldsymbol{\Lambda})\right)}{\partial \theta} \\ &+ \frac{\partial}{\partial \theta} \log \mathcal{N}\left(\widetilde{\mathbf{m}} \,\middle|\, 0, \widetilde{\mathbf{K}}\right),\end{aligned}$$

where with Proposition 14 on page 61,

$$\frac{\partial}{\partial \theta} \log \mathcal{N}\left(\widetilde{\mathbf{m}} \,\middle|\, 0, \widetilde{\mathbf{K}}\right) = -\frac{1}{2}\left(\widetilde{\mathbf{m}}^T \widetilde{\mathbf{K}}^{-1}\left(2\frac{\partial \widetilde{\mathbf{m}}}{\partial \theta} - \frac{\partial \widetilde{\mathbf{K}}}{\partial \theta} \widetilde{\mathbf{K}}^{-1} \widetilde{\mathbf{m}}\right) + \operatorname{tr}\left(\widetilde{\mathbf{K}}^{-1} \frac{\partial \widetilde{\mathbf{K}}}{\partial \theta}\right)\right).$$

Applying Proposition 15 on page 61, we have

$$\begin{aligned}\frac{\partial \log\left(|\beta \boldsymbol{\Lambda}_l| \mathcal{N}(\beta \mathbf{r}_l \mid 0, \beta \boldsymbol{\Lambda}_l)\right)}{\partial \theta} &= \frac{1}{2}\left(\beta \mathbf{r}_l^T \boldsymbol{\Lambda}_l^{-1}\left(\frac{\partial \boldsymbol{\Lambda}_l}{\partial \theta} \boldsymbol{\Lambda}_l^{-1} \mathbf{r}_l - 2\frac{\partial \mathbf{r}_l}{\partial \theta}\right) + \operatorname{tr}\left(\boldsymbol{\Lambda}_l^{-1} \frac{\partial \boldsymbol{\Lambda}_l}{\partial \theta}\right)\right), \\ \frac{\partial \log\left(|\boldsymbol{\Lambda}| \mathcal{N}(\mathbf{r} \mid 0, \boldsymbol{\Lambda})\right)}{\partial \theta} &= \frac{1}{2}\left(\mathbf{r}^T \boldsymbol{\Lambda}^{-1}\left(\frac{\partial \boldsymbol{\Lambda}}{\partial \theta} \boldsymbol{\Lambda}^{-1} \mathbf{r} - 2\frac{\partial \mathbf{r}}{\partial \theta}\right) + \operatorname{tr}\left(\boldsymbol{\Lambda}^{-1} \frac{\partial \boldsymbol{\Lambda}}{\partial \theta}\right)\right),\end{aligned}$$

where

$$\begin{aligned}\frac{\partial \mathbf{r}_l}{\partial \theta} &= \frac{\partial \mathbf{A}_l}{\partial \theta} \boldsymbol{\Sigma}_l^{-1} \boldsymbol{\mu}_l + \mathbf{A}_l \boldsymbol{\Sigma}_l^{-1}\left(\frac{\partial \boldsymbol{\mu}_l}{\partial \theta} - \frac{\partial \boldsymbol{\Sigma}_l}{\partial \theta} \boldsymbol{\Sigma}_l^{-1} \boldsymbol{\mu}_l\right), \\ \frac{\partial \boldsymbol{\Lambda}_l}{\partial \theta} &= \frac{\partial \mathbf{A}_l}{\partial \theta} \boldsymbol{\Sigma}_l^{-1} \mathbf{A}_l^T + \mathbf{A}_l \boldsymbol{\Sigma}_l^{-1}\left(\frac{\partial \mathbf{A}_l^T}{\partial \theta} - \frac{\partial \boldsymbol{\Sigma}_l}{\partial \theta} \boldsymbol{\Sigma}_l^{-1} \mathbf{A}_l^T\right), \\ \frac{\partial \mathbf{r}}{\partial \theta} &= \beta \sum_{l=1}^{L} \frac{\partial \mathbf{r}_l}{\partial \theta} + \widetilde{\mathbf{K}}^{-1}\left(\frac{\partial \widetilde{\mathbf{m}}}{\partial \theta} - \frac{\partial \widetilde{\mathbf{K}}}{\partial \theta} \widetilde{\mathbf{K}}^{-1} \widetilde{\mathbf{m}}\right), \\ \frac{\partial \boldsymbol{\Lambda}}{\partial \theta} &= \beta \sum_{l=1}^{L} \frac{\partial \boldsymbol{\Lambda}_l}{\partial \theta} - \widetilde{\mathbf{K}}^{-1} \frac{\partial \widetilde{\mathbf{K}}}{\partial \theta} \widetilde{\mathbf{K}}^{-1}.\end{aligned}$$

Finally, we infer

$$\begin{aligned}
\frac{\partial \mathbf{A}_l}{\partial \theta} &= \widetilde{\mathbf{K}}^{-1}\left(\frac{\partial \widetilde{\mathbf{K}}_l^T}{\partial \theta} - \frac{\partial \widetilde{\mathbf{K}}}{\partial \theta}\mathbf{A}_l\right), \\
\frac{\partial \boldsymbol{\mu}_l}{\partial \theta} &= \frac{\partial \mathbf{A}_l^T}{\partial \theta}\widetilde{\mathbf{m}} + \mathbf{A}_l^T\frac{\partial \widetilde{\mathbf{m}}}{\partial \theta} - \frac{\partial \mathbf{m}_l}{\partial \theta}, \\
\frac{\partial \boldsymbol{\Sigma}_l}{\partial \theta} &= \frac{\partial \mathbf{K}_l}{\partial \theta} - \frac{\partial \widetilde{\mathbf{K}}_l}{\partial \theta}\mathbf{A}_l - \widetilde{\mathbf{K}}_l\frac{\partial \mathbf{A}_l}{\partial \theta}.
\end{aligned}$$

4.3.2. β-Noise Approximation Set Coding

Gaussian process regression always has the noise standard deviation σ_n built-in. For this reason, the role of β is reflected in σ_n to obtain the variant of β-noise approximation set coding. As mentioned earlier, it can be recovered by fixing $\beta = 1$ in Boltzmann approximation set coding.

4.3.3. Bayesian Approximation Set Coding

To calculate the posterior of Gaussian process regression, first note that

$$\begin{bmatrix} \widetilde{\mathbf{f}} \\ \mathbf{y}_l \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \widetilde{\mathbf{m}} \\ \mathbf{m}_l \end{bmatrix}, \begin{bmatrix} \widetilde{\mathbf{K}} & \widetilde{\mathbf{K}}_l^T \\ \widetilde{\mathbf{K}}_l & \mathbf{K}_l \end{bmatrix}\right).$$

For this Gaussian to be non-degenerate, the objects $\widetilde{\mathbf{X}}$ need to be chosen such that $\widetilde{\mathbf{K}}$ is invertible, like in the other variants. Given Equation (2.4) on page 12, the posterior evaluated at $\widetilde{\mathbf{X}}$ is

$$\widetilde{\mathbf{f}} \,\Big|\, \mathbf{X}, \mathbf{y}_l, \widetilde{\mathbf{X}} \sim \mathcal{N}(\mathbf{s}_l, \mathbf{V}_l),$$

where

$$\begin{aligned}
\mathbf{s}_l &= \widetilde{\mathbf{m}} + \mathbf{B}_l^T(\mathbf{y}_l - \mathbf{m}_l) \in \mathbb{R}^M, \\
\mathbf{V}_l &= \widetilde{\mathbf{K}} - \widetilde{\mathbf{K}}_l^T\mathbf{B}_l \in \mathbb{R}^{M\times M}, \\
\mathbf{B}_l &= \mathbf{K}_l^{-1}\widetilde{\mathbf{K}}_l \in \mathbb{R}^{N_l\times M}.
\end{aligned}$$

According to Equation (3.2) on page 25, the posterior agreement in its raw form is then

$$\eta = \int_{\mathbb{R}^M}\left(\prod_{l=1}^{L}\mathcal{N}\left(\widetilde{\mathbf{f}} \,\Big|\, \mathbf{s}_l, \mathbf{V}_l\right)\right)\mathcal{N}\left(\widetilde{\mathbf{f}} \,\Big|\, \widetilde{\mathbf{K}}, \widetilde{\mathbf{m}}\right)\, d^M\widetilde{\mathbf{f}}.$$

Reusing Proposition 11 on page 27, we have

$$\eta = \frac{\prod_{l=1}^{L}\mathcal{N}(\mathbf{s}_l \mid \mathbf{0}, \mathbf{V}_l)}{|\mathbf{P}|\,\mathcal{N}(\mathbf{s} \mid \mathbf{0}, \mathbf{P})}\mathcal{N}\left(\widetilde{\mathbf{m}} \,\Big|\, \mathbf{0}, \widetilde{\mathbf{K}}\right), \tag{4.4}$$

where

$$\mathbf{s} = \sum_{l=1}^{L}\mathbf{V}_l^{-1}\mathbf{s}_l + \widetilde{\mathbf{K}}^{-1}\widetilde{\mathbf{m}}, \qquad \mathbf{P} = \sum_{l=1}^{L}\mathbf{V}_l^{-1} + \widetilde{\mathbf{K}}^{-1}.$$

Gradient

The derivative with respect to an arbitrary hyperparameter θ is

$$\frac{\partial \log \eta_\beta}{\partial\theta} = \sum_{l=1}^{L} \frac{\partial \log \mathcal{N}(\mathbf{s}_l \mid \mathbf{0}, \mathbf{V}_l)}{\partial\theta} - \frac{\partial \log (|\mathbf{P}| \mathcal{N}(\mathbf{s} \mid \mathbf{0}, \mathbf{P}))}{\partial\theta} + \frac{\partial}{\partial\theta} \log \mathcal{N}\left(\tilde{\mathbf{m}} \,\middle|\, \mathbf{0}, \tilde{\mathbf{K}}\right).$$

By Proposition 14 on page 61,

$$\frac{\partial \log \mathcal{N}(\mathbf{s}_l \mid \mathbf{0}, \mathbf{V}_l)}{\partial\theta} = -\frac{1}{2}\left(\mathbf{s}_l^{\mathsf{T}} \mathbf{V}_l^{-1}\left(2\frac{\partial \mathbf{s}_l}{\partial\theta} - \frac{\partial \mathbf{V}_l}{\partial\theta}\mathbf{V}_l^{-1}\mathbf{s}_l\right) + \operatorname{tr}\left(\mathbf{V}_l^{-1}\frac{\partial \mathbf{V}_l}{\partial\theta}\right)\right),$$

$$\frac{\partial}{\partial\theta}\log \mathcal{N}\left(\tilde{\mathbf{m}} \,\middle|\, \mathbf{0}, \tilde{\mathbf{K}}\right) = -\frac{1}{2}\left(\tilde{\mathbf{m}}^{\mathsf{T}} \tilde{\mathbf{K}}^{-1}\left(2\frac{\partial \tilde{\mathbf{m}}}{\partial\theta} - \frac{\partial \tilde{\mathbf{K}}}{\partial\theta}\tilde{\mathbf{K}}^{-1}\tilde{\mathbf{m}}\right) + \operatorname{tr}\left(\tilde{\mathbf{K}}^{-1}\frac{\partial \tilde{\mathbf{K}}}{\partial\theta}\right)\right),$$

and by Proposition 15 on page 61,

$$\frac{\partial \log (|\mathbf{P}| \mathcal{N}(\mathbf{s} \mid \mathbf{0}, \mathbf{P}))}{\partial\theta} = \frac{1}{2}\left(\mathbf{s}^{\mathsf{T}} \mathbf{P}^{-1}\left(\frac{\partial \mathbf{P}}{\partial\theta}\mathbf{P}^{-1}\mathbf{s} - 2\frac{\partial \mathbf{s}}{\partial\theta}\right) + \operatorname{tr}\left(\mathbf{P}^{-1}\frac{\partial \mathbf{P}}{\partial\theta}\right)\right).$$

We further have

$$\frac{\partial \mathbf{s}}{\partial\theta} = \sum_{l=1}^{L} \mathbf{V}_l^{-1}\left(\frac{\partial \mathbf{s}_l}{\partial\theta} - \frac{\partial \mathbf{V}_l}{\partial\theta}\mathbf{V}_l^{-1}\mathbf{s}_l\right) + \tilde{\mathbf{K}}^{-1}\left(\frac{\partial \tilde{\mathbf{m}}}{\partial\theta} - \frac{\partial \tilde{\mathbf{K}}}{\partial\theta}\tilde{\mathbf{K}}^{-1}\tilde{\mathbf{m}}\right),$$

$$\frac{\partial \mathbf{P}}{\partial\theta} = -\sum_{l=1}^{L} \mathbf{V}_l^{-1}\frac{\partial \mathbf{V}_l}{\partial\theta}\mathbf{V}_l^{-1} - \tilde{\mathbf{K}}^{-1}\frac{\partial \tilde{\mathbf{K}}}{\partial\theta}\tilde{\mathbf{K}}^{-1},$$

and

$$\frac{\partial \mathbf{s}_l}{\partial\theta} = \frac{\partial \tilde{\mathbf{m}}}{\partial\theta} + \frac{\partial \mathbf{B}_l^{\mathsf{T}}}{\partial\theta}(\mathbf{y}_l - \mathbf{m}_l) - \mathbf{B}_l^{\mathsf{T}}\frac{\partial \mathbf{m}_l}{\partial\theta},$$

$$\frac{\partial \mathbf{V}_l}{\partial\theta} = \frac{\partial \tilde{\mathbf{K}}}{\partial\theta} - \frac{\partial \tilde{\mathbf{K}}_l^{\mathsf{T}}}{\partial\theta}\mathbf{B}_l - \tilde{\mathbf{K}}_l^{\mathsf{T}}\frac{\partial \mathbf{B}_l}{\partial\theta},$$

$$\frac{\partial \mathbf{B}_l}{\partial\theta} = \mathbf{K}_l^{-1}\left(\frac{\partial \tilde{\mathbf{K}}_l}{\partial\theta} - \frac{\partial \mathbf{K}_l}{\partial\theta}\mathbf{B}_l\right).$$

4.4. Implementation

Let us look at important aspects of a possible implementation.

4.4.1. Asymptotic Time Complexity

First of all, Table 4.1 on the next page gives an overview of the asymptotic time complexity to evaluate I_β^θ once. Matrix computations dominate the complexity. We assume

Variants	Decompositions		Linear systems		Multiplications	
Boltzmann ASC	$\widetilde{\mathbf{K}}$	$\Theta\left(M^3\right)$	$\widetilde{\mathbf{K}}^{-1}\widetilde{\mathbf{K}}_l^T$	$\Theta\left(M^2 N_l\right)$	$\widetilde{\mathbf{K}}_l \mathbf{A}_l$	$\Theta\left(M N_l^2\right)$
β-noise ASC	$\mathbf{\Sigma}_l$	$\Theta\left(N_l^3\right)$	$\mathbf{\Sigma}_l^{-1}\mathbf{A}_l^T$	$\Theta\left(M N_l^2\right)$	$\mathbf{A}_l\left(\mathbf{\Sigma}_l^{-1}\mathbf{A}_l^T\right)$	$\Theta\left(M^2 N_l\right)$
	$\beta\mathbf{\Lambda}_l$	$\Theta\left(M^3\right)$				
	$\mathbf{\Lambda}$	$\Theta\left(M^3\right)$				
Bayesian ASC	$\widetilde{\mathbf{K}}$	$\Theta\left(M^3\right)$	$\mathbf{K}_l^{-1}\widetilde{\mathbf{K}}_l$	$\Theta\left(M N_l^2\right)$	$\widetilde{\mathbf{K}}_l^T \mathbf{B}_l$	$\Theta\left(M^2 N_l\right)$
	$\mathbf{K}_l$	$\Theta\left(N_l^3\right)$				
	$\mathbf{V}_l$	$\Theta\left(M^3\right)$				
	$\mathbf{P}$	$\Theta\left(M^3\right)$				

Table 4.1.: Asymptotic time complexity of the dominant computations to evaluate the mutual information I_β^θ of approximation set coding (ASC) once: matrix decompositions, solving linear systems of equations with multiple right-hand sides, and matrix multiplications.

standard algorithms for the Cholesky decomposition, solving a linear system of equations with such a decomposition, and matrix multiplications. Based on this analysis, the time for one such evaluation is asymptotically the same for every criterion, namely

$$\Theta\left(\sum_{l=1}^{L}\left(N_l^3 + M N_l^2 + M^2 N_l\right) + M^3\right). \tag{4.5}$$

For instance, a realistic scenario could be a constant number $L \in \Theta(1)$ of subsets with $M, N_l \in \Theta(N)$. The evaluation time would then be $\Theta\left(N^3\right)$, on a par with the criteria of maximum evidence or leave-one-out cross-validation. When averaging the approximation capacity of J partitions as in Algorithm 2 on page 31, the time grows to $\Theta\left(JN^3\right)$.

However, this is only the case for a fixed approximation precision β. Boltzmann approximation set coding optimizes with respect to β for each fixed θ, which multiplies its total number of evaluations. In stark contrast, β-noise and Bayesian approximation set coding do not add β to the optimization, making them significantly faster.

4.4.2. Issues with Numerically Singular Matrices

In theory, only noise-free covariance matrices $\mathbf{K}_f$ can be singular, but not so the covariance matrices $\mathbf{K}_y$ due to the added $\sigma_n^2\mathbf{I}$. Yet in practice, it happens more than one might expect that either kind of matrices is numerically considered singular, triggered by a failing Cholesky decomposition. This occurs because the eigenvalues of a covariance matrix "can decay very rapidly." [30, page 201] Out of the J random partitions, we exclude those where failures like this happen from the average.

4.4.3. Numerical Software Used

The specific implementation of this thesis uses of a number of software libraries for its numerical processing. Their authors deserve proper credit for their brilliant work. Developed in Scala, the implementation relies on Breeze [16] for linear algebra, random number generation, and the optimization algorithm of limited-memory BFGS [27]. Efficient LAPACK routines related to the Cholesky decomposition are invoked through the wrapper netlib-java [17]. Regarding computations with Gaussian processes, the covariance functions of the GPML toolbox [29] are reused, calling them via matlabcontrol [20].

5. Results

The twofold aim of model selection is hyperparameter optimization and functional form selection, according to which we design our experiments. We compare the variants of approximation set coding (ASC) with each other and with the classics of maximum evidence and leave-one-out (LOO) cross-validation.

5.1. Setup

On the one hand, we generate synthetic data sets drawn from Gaussian processes with noise σ_n. More precisely, for random inputs $\mathbf{X}$, their outputs $\mathbf{y}$ are drawn from the evidence distribution of Equation (2.3) on page 12. The matrix entries of the inputs $\mathbf{X}$ are independent and identically distributed with a standard Gaussian distribution as in $x_{d,n} \sim \mathcal{N}(0, 1)$. This way, the inputs are concentrated around the origin, but are occasionally still farer away from it for a wider spread. The generative model is also known as the teacher, while the student is the learned model. A misspecification is thus described by a situation where the teacher and student have different functional forms.

Besides synthetic data, we work with a simple real-world data set to check against an instance with the typical complexity of nature. Berkeley Earth[1] provides a data set with Earth's daily average temperature from 1880 until 2014 on the land surface, given as a difference from 8.68 °C (anomaly) [3]. From this, our real-world data is aggregated as the mean anomaly per day of the year, shown in Figure 5.1 on the following page. It has $N = 365$ cases in $D = 1$ dimension.

We always fix the mean function of Gaussian processes to the zero mean function. In contrast, the functional form of the covariance functions is varied. We focus on the covariance functions of Figure 2.1 on page 13, of which the squared exponential covariance function is given most attention due to its popularity [13].

The standardized mean squared error serves as a measure of how good predictions are. If the predictive mean is $\bar{\mathbf{f}} \in \mathbb{R}^N$, but the actual outputs are $\mathbf{y}$, then the standardized mean squared error equals

$$\frac{\|\mathbf{y} - \bar{\mathbf{f}}\|_2^2}{N\sigma_y^2}, \tag{5.1}$$

where σ_y^2 is the sample variance of $\mathbf{y}$. The predictive distribution for Gaussian process regression is Equation (2.4) on page 12.

[1]http://www.berkeleyearth.org

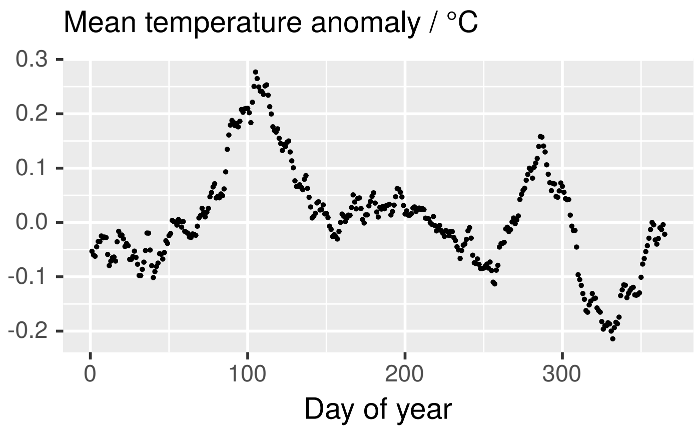


Figure 5.1.: Earth's land temperature averaged per day of the year from 1880 until 2014, given as a difference from 8.68 °C (anomaly) [3].

The visualizations are created with ggplot2 [35]. For certain plots, we visualize the mean of a sample, plus and minus one standard deviation. Other plots indicate confidence intervals estimated by the basic nonparametric bootstrap not assuming normality.

5.1.1. Optimization Algorithm

Any objective function is numerically optimized based on the same setup. For variables that are constrained to be positive such as σ_n or β, we optimize with respect to their natural logarithm to make the optimization unconstrained. The algorithm of limited-memory BFGS [27] is configured with memory $m = 4$ and a relative tolerance of 10^{-7}. For an initial guess of an optimization run, we randomly draw from the standard Cauchy distribution due to its fat tails. How should the number of iterations be limited?

Experiment 1. From a Gaussian process as a teacher model, we randomly draw 256 data sets, each of dimension $D = 1$ with $N = 64$ cases. The student model has the same functional form as the teacher model, but does not know any of its hyperparameters. For a given criterion, we run the optimization algorithm once, without limiting the number of its iterations. If the optimization succeeds, we remember the number of iterations to get there. See Figure 5.2 on the next page.

Based on these samples summarized in Figure 5.2 on the facing page, we decide to configure the optimization algorithm with a maximum of 16 iterations. While this looks quite tight, we repeat an optimization several times, as an objective function can have many local optima. In fact, we do 16 runs per optimization problem and take the solution with the best value. If all of these optimization runs fail for a criterion applied to a certain data set, we still count the data set instead of drawing a fresh one.

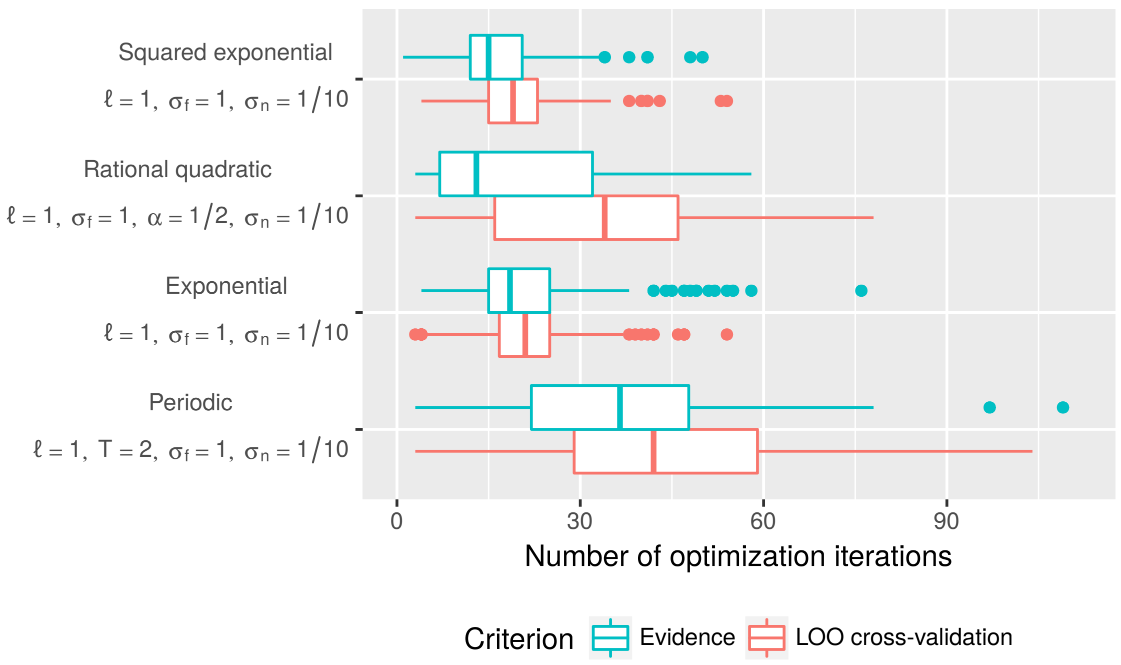


Figure 5.2.: Number of optimization iterations according to Experiment 1 on page 40 for various criteria and covariance functions.

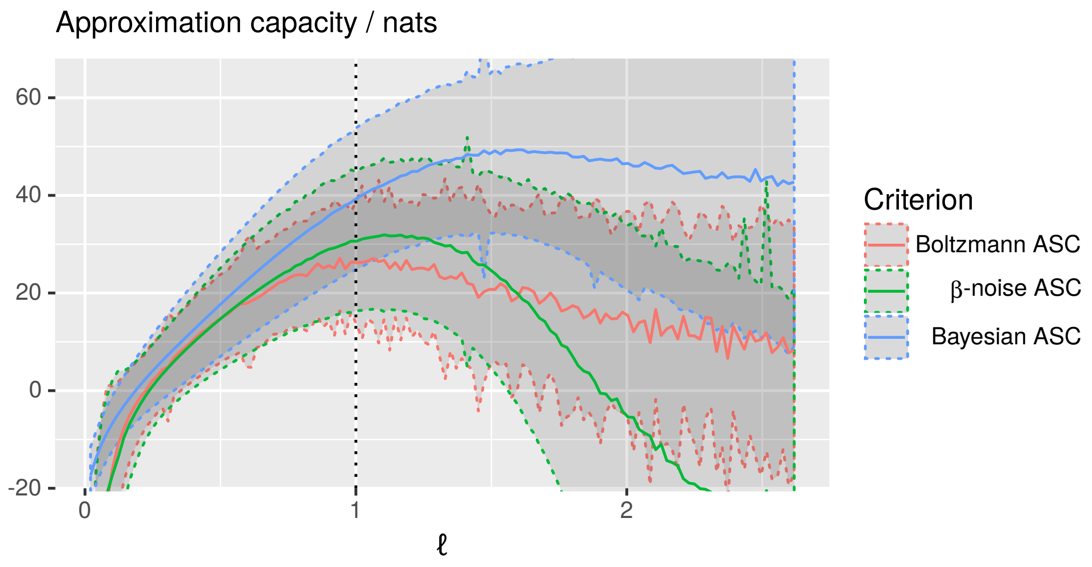
5.2. Hyperparameter Optimization

First of all, we examine the objective functions of approximation set coding. It is interesting to see where their optima lie in relation to the hyperparameters with which the synthetic data is generated. We focus on the squared exponential covariance function.

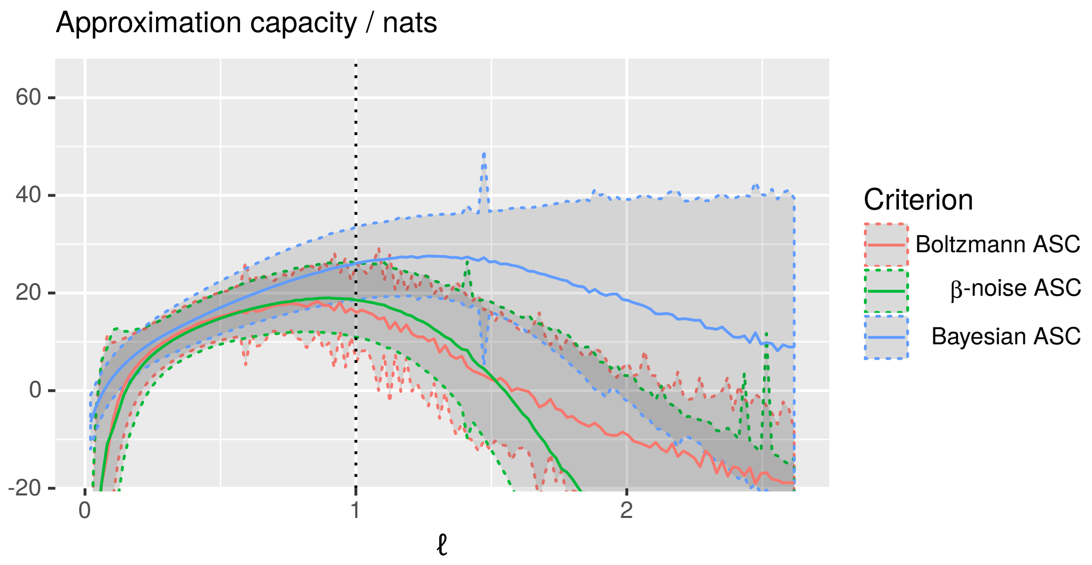
Experiment 2. From a Gaussian process as a teacher model, we randomly draw 256 data sets, each of dimension $D = 1$ with $N = 64$ cases. The student model is the same as the teacher model, except that it has one unknown hyperparameter θ at a time. For a given criterion, we draw one objective function per data set. Approximation set coding is done with $L = 2$ subsets and $M = 8$ objects. The θ-axis is discretized by 128 equally spaced points. To remove outliers caused by numerical issues, the statistics solely consider approximation capacities over -128 nats. See Figures 5.3 to 5.5 on pages 42–44.

Looking at the optima, β-noise approximation set coding with $h(\tau) = 0$ seems to perform well in every case. Boltzmann approximation set coding has similar optima, except that its objective functions for the hyperparameter σ_n appear quite flat. For Bayesian approximation set coding, an optimum exists each time, which is good in case of σ_n but not otherwise. So far, there is no clear favorite.

We now optimize the objective functions to do hyperparameter optimization. The criteria are then compared on the test error for a test data set drawn from the same

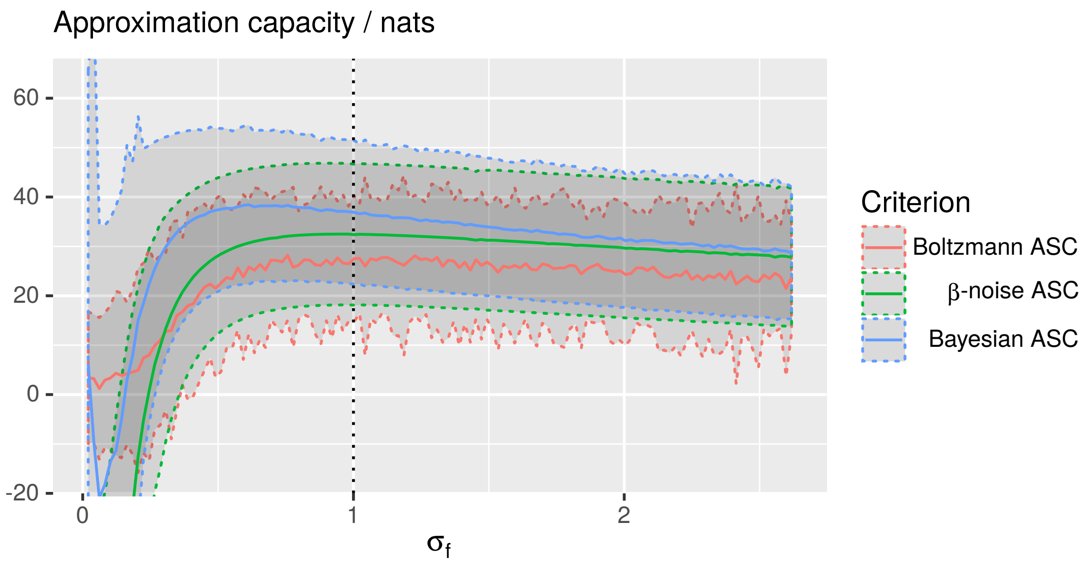


(a) $h(\tau) = 0$.

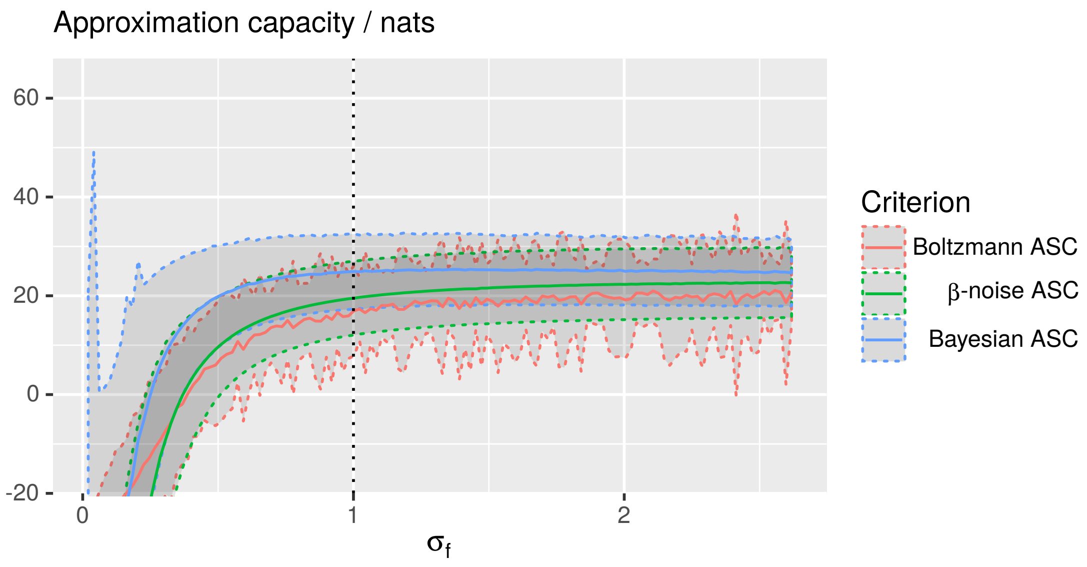


(b) $h(\tau) = h\left(\mathcal{N}\left(\widetilde{m}, \widetilde{K}\right)\right)$.

Figure 5.3.: Mean of objective functions drawn according to Experiment 2 on page 41 for the squared exponential covariance function where $\sigma_f = 1$ and $\sigma_n = 1/10$. The generative model additionally fixes $\ell = 1$.

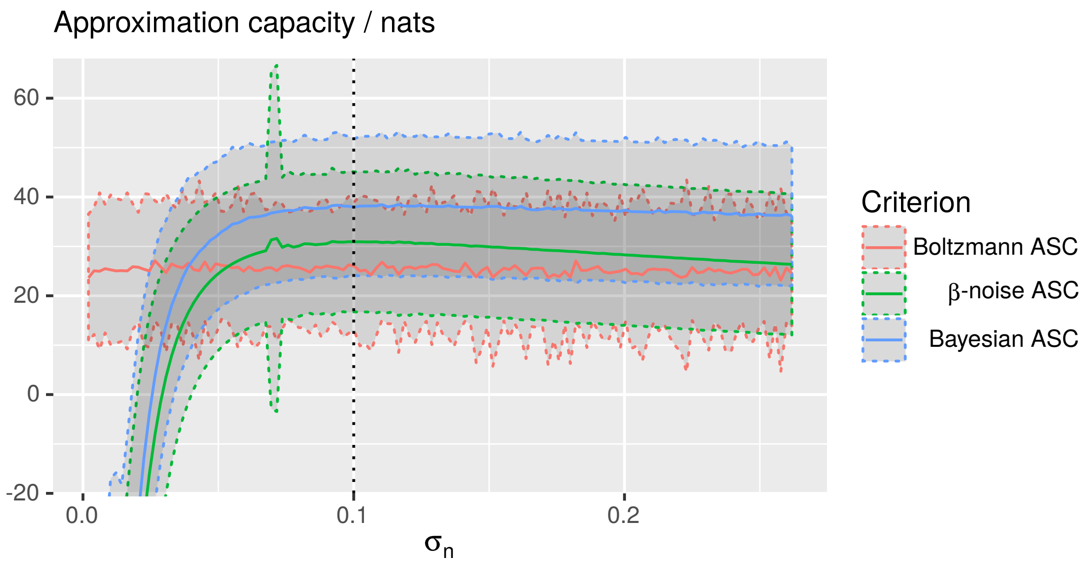


(a) $h(\tau) = 0$.

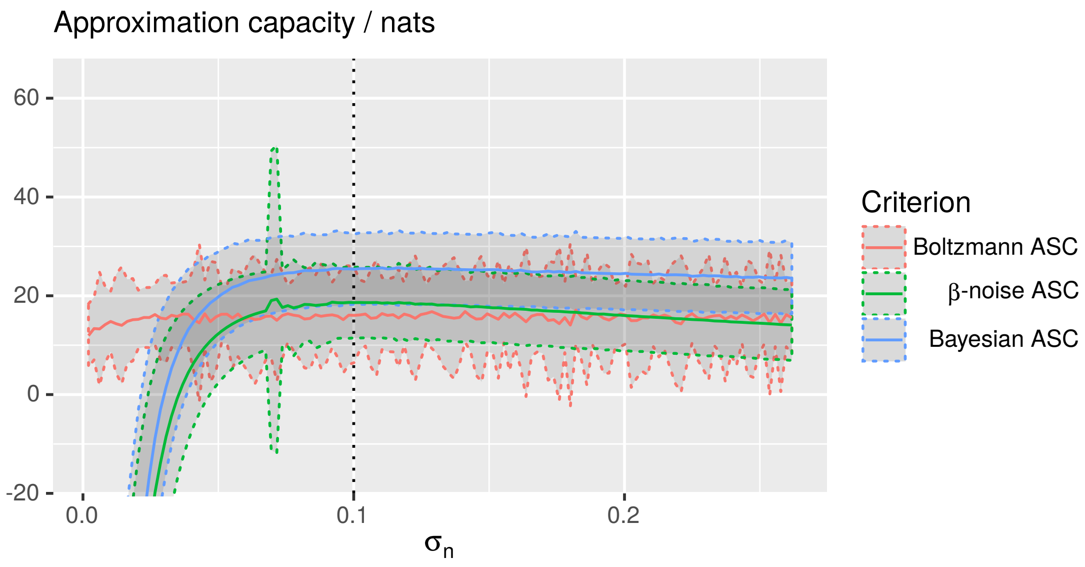


(b) $h(\tau) = h\left(\mathcal{N}\left(\tilde{m}, \tilde{K}\right)\right)$.

Figure 5.4.: Mean of objective functions drawn according to Experiment 2 on page 41 for the squared exponential covariance function where $\ell = 1$ and $\sigma_n = 1/10$. The generative model additionally fixes $\sigma_f = 1$.



(a) $h(\tau) = 0$.



(b) $h(\tau) = h\left(\mathcal{N}\left(\widetilde{m}, \widetilde{K}\right)\right)$.

Figure 5.5.: Mean of objective functions drawn according to Experiment 2 on page 41 for the squared exponential covariance function where $\ell = 1$ and $\sigma_f = 1$. The generative model additionally fixes $\sigma_n = 1/10$.

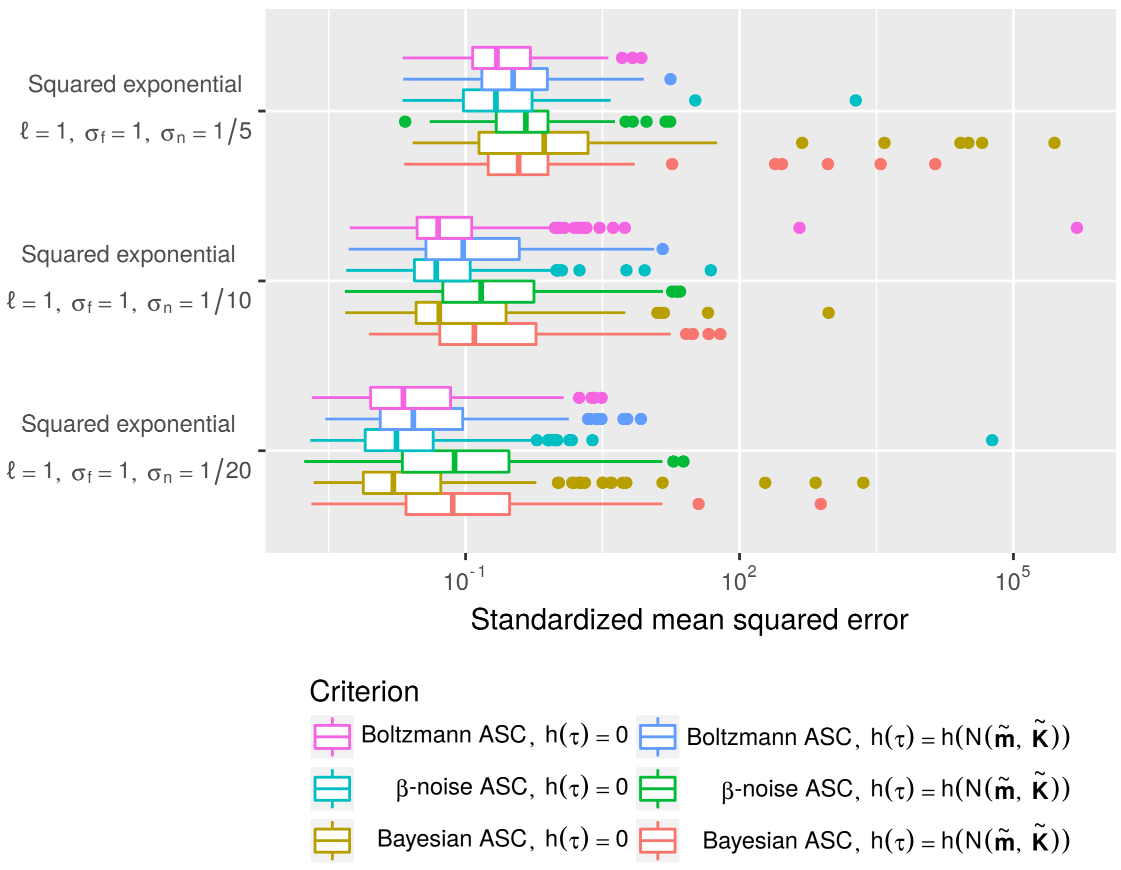


Figure 5.6.: Test errors for the variants of approximation set coding according to Experiment 3.

generative model.

Experiment 3. From a Gaussian process as a teacher model, we randomly draw 256 data sets, each of dimension $D = 1$ with $N = 64$ training and 2048 test cases. The student model has the same functional form as the teacher model, but does not know any of its hyperparameters. It selects hyperparameters based on the training data set. Approximation set coding is done with $J = 32$ partitions, $L = 2$ subsets, and $M = 4$ objects. The standardized mean squared error on the test data set is reported. See Figures 5.6 and 5.7 on the current page and on the following page.

Figure 5.6 suggests that Boltzmann and β-noise approximation set coding with $h(\tau) = 0$ have the best performance. For little noise σ_n, Bayesian approximation set coding with $h(\tau) = 0$ has a competitive median, but there exist more outliers in general. Figure 5.7 on the following page then compares those winners of approximation set coding to the classics. There is no doubt that maximum evidence is strongest here. Leave-one-out

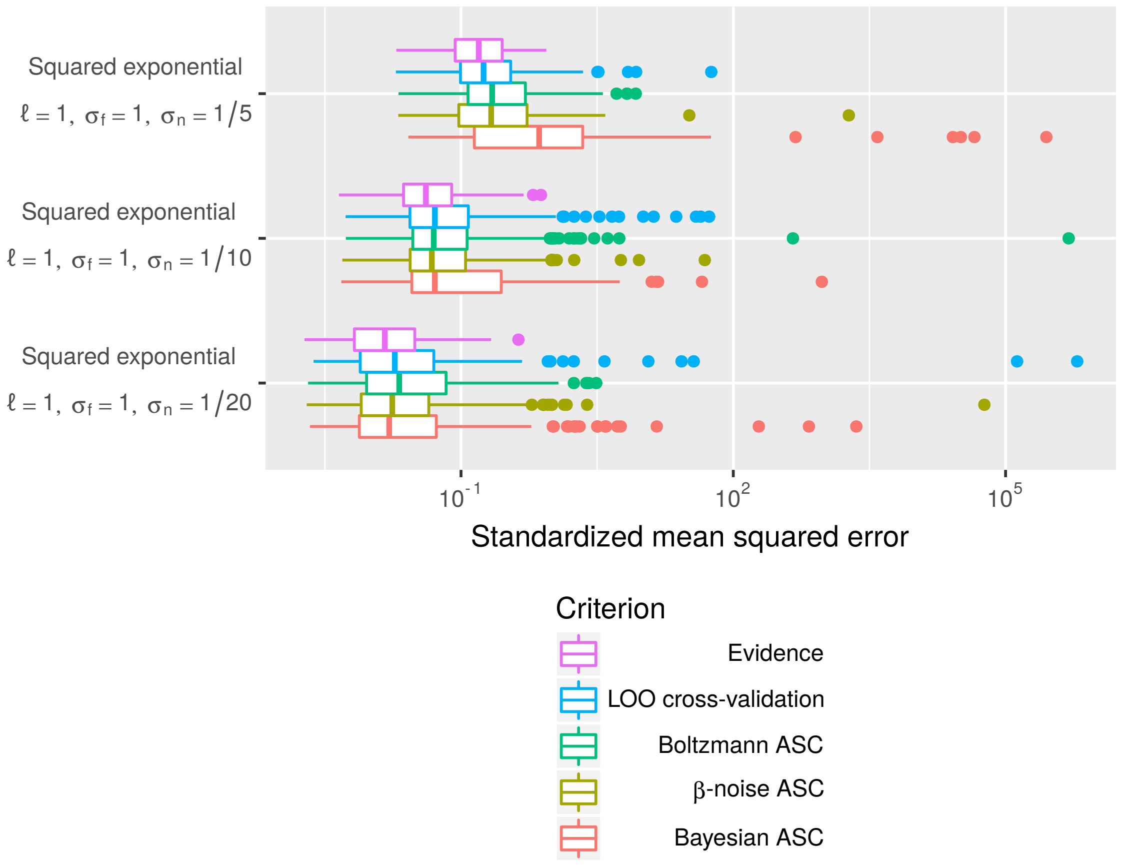


Figure 5.7.: Test errors for approximation set coding with $h(\tau) = 0$ compared with the classics according to Experiment 3 on page 45.

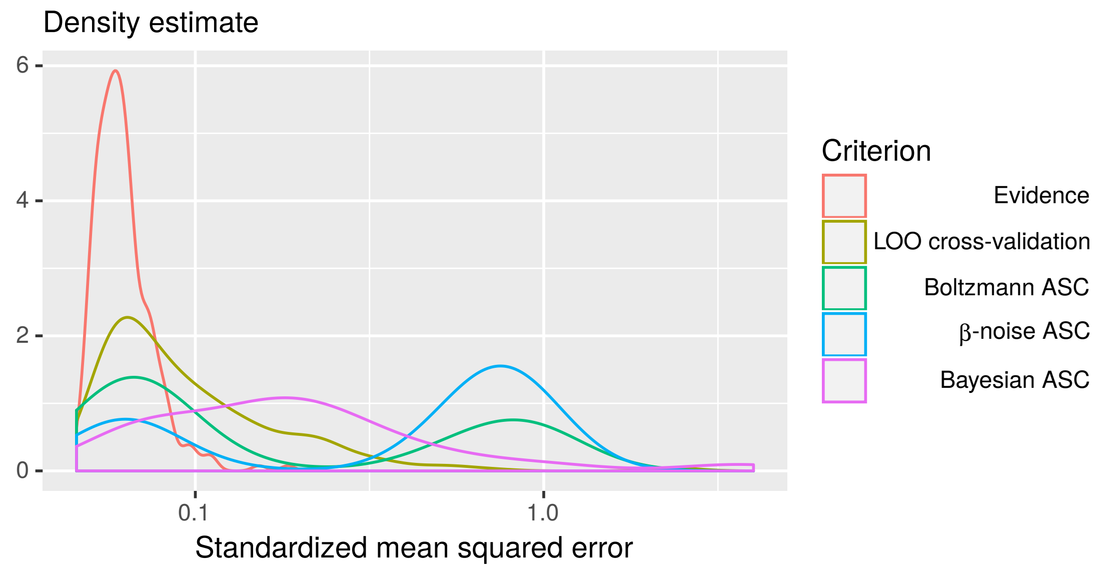


Figure 5.8.: Test errors on real-world data according to Experiment 4 for the squared exponential covariance function. Approximation set coding is done with $h(\tau) = 0$.

cross-validation is good, too, but has many outliers compared to the other criteria. β-noise approximation set coding seems compete with it, especially for little noise.

We now turn to the real-world data set described in Section 5.1 on page 39.

Experiment 4. We randomly partition our real-world data 256 times into 64 training and 301 test cases. Each time, the hyperparameters are then optimized on the training data set to record the standardized mean squared error on the test data set. Approximation set coding is done with $J = 32$ partitions, $L = 2$ subsets, and $M = 8$ objects. See Figure 5.8.

The criterion of maximum evidence works best again, with the distribution of its test errors being very peaked at a level unreached by any other criterion. Leave-one-out cross-validation comes second with a higher variance. Boltzmann approximation set coding has a similar mode, but a second peak at a significant test error of almost 1 causes trouble. Clearly, β-noise approximation set coding is worse. Bayesian approximation set coding is widely spread between the two peaks with a couple of outliers.

Next, we would like to see if approximation set coding could help to decide between two optimized hyperparameter vectors: one estimated by maximum evidence and the other by leave-one-out cross-validation.

Experiment 5. The real-world data is repeatedly partitioned into 64 training and 301 test cases. We optimize the hyperparameters once with maximum evidence and another time with leave-one-out cross-validation to get $\theta^{\star}_{ME}$ and $\theta^{\star}_{CV}$, respectively. Then the standardized mean squared error on the test data set is computed for $\theta^{\star}_{ME}$ and $\theta^{\star}_{CV}$,

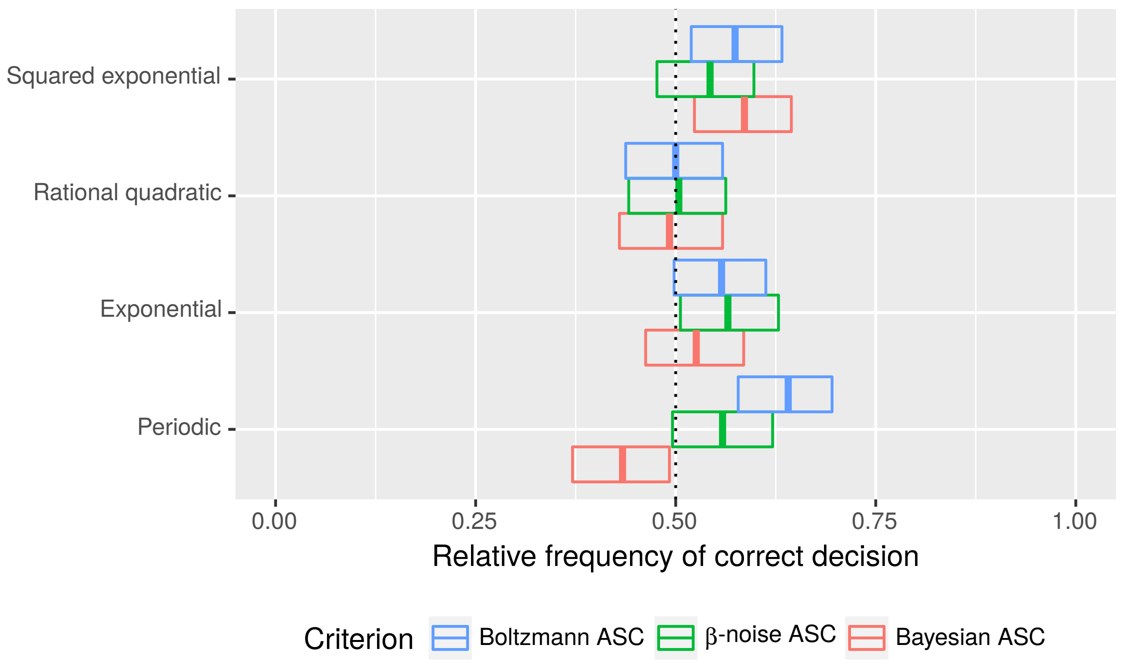


Figure 5.9.: Chances that approximation set coding with $h(\tau) = 0$ correctly decides between maximum evidence and leave-one-out cross-validation according to Experiment 5 on page 47. The mean is visualized with a 95 % confidence interval.

which ranks them. We compare this ranking with the ranking induced by the approximation capacity. If these two rankings are the same, that is, if the hyperparameters θ with a lower test error have a higher approximation capacity, we count it as a correct decision, otherwise not. The number of random partitions is 256, where for exactly half of them, $\theta^{\star}_{ME}$ has a lower test error than $\theta^{\star}_{CV}$, and vice versa for the other half. Approximation set coding is done with $J = 32$ partitions, $L = 2$ subsets, and $M = 8$ objects. See Figure 5.9.

Overall, Boltzmann approximation set coding appears to help most with making a decision, giving the top improvement for the periodic covariance function. Bayesian approximation set coding is best in case of the squared exponential covariance function, while it can worsen the decision for the periodic covariance function. If any at all, β-noise approximation causes gentle improvements.

5.3. Functional Form Selection

We proceed to compare the criteria on function form selection. Regarding synthetic data, when a criterion is given a number of functional forms to select from, it should select the one the generative model is based on. The game we play is thus to let a criterion rank a number of student functional forms, of which exactly one belongs to the teacher model. The ranking is simply induced by the cost function of the respective criterion. In order to separate concerns, we optimize the hyperparameters by a fixed criterion. Leave-one-out cross-validation is first assigned this role, as it should be more resistant to model misspecification than maximum evidence [2].

Experiment 6. From a Gaussian process as a teacher model, we randomly draw 256 data sets, each of dimension $D = 1$ with $N = 128$ cases. For every data set, we optimize the hyperparameters of a number of student functional forms by leave-one-out cross-validation, not revealing any information about the teacher model. We then plug the resulting student models into the cost functions of various criteria, which gives rise to a ranking. The student model of rank 1 has the lowest costs, meaning that it would be selected. Approximation set coding is done with $J = 32$ partitions, $L = 2$ subsets, and $M = 8$ objects. See Figure 5.10 on the following page.

The criterion of maximum evidence has again the best performance, selecting the right model each time. This might be surprising, given that in other misspecification scenarios, it appears to do worse than cross-validation [2]. Overall, leave-one-out cross-validation does quite well, but there is a chance that it could confuse a squared exponential and a rational quadratic for a periodic covariance function. Bayesian approximation set coding with $h(\tau) = 0$ performs not bad, either. Except that on average it confuses a rational quadratic for a squared exponential covariance function, it almost unambiguously selects the right model otherwise. The remaining variants of approximation set coding, including those with $h(\tau) \neq 0$ not shown here, seem not to work well for functional form selection.

How do things change if we choose the criterion of maximum evidence for hyperparameter optimization?

Experiment 7. This is the same as Experiment 6, except that the hyperparameters are optimized by maximum evidence instead of leave-one-out cross-validation. See Figure 5.11 on page 51.

No criteria does really become better. On the contrary, maximum evidence could now possibly confuse a squared exponential for a rational quadratic covariance function. Other than that, it is again the leader. Leave-one-out cross-validation stays about the same. Approximation set coding is still not really good, with Bayesian approximation set coding even confusing the exponential for the periodic covariance function.

We would next like to see functional form selection on the real-world data set described in Section 5.1 on page 39. As the teacher model is unknown, we compare to the test error on separate test data.

Experiment 8. We randomly partition our real-world data 256 times into 64 training and 301 test cases. Given a training data set, the hyperparameters are optimized for a

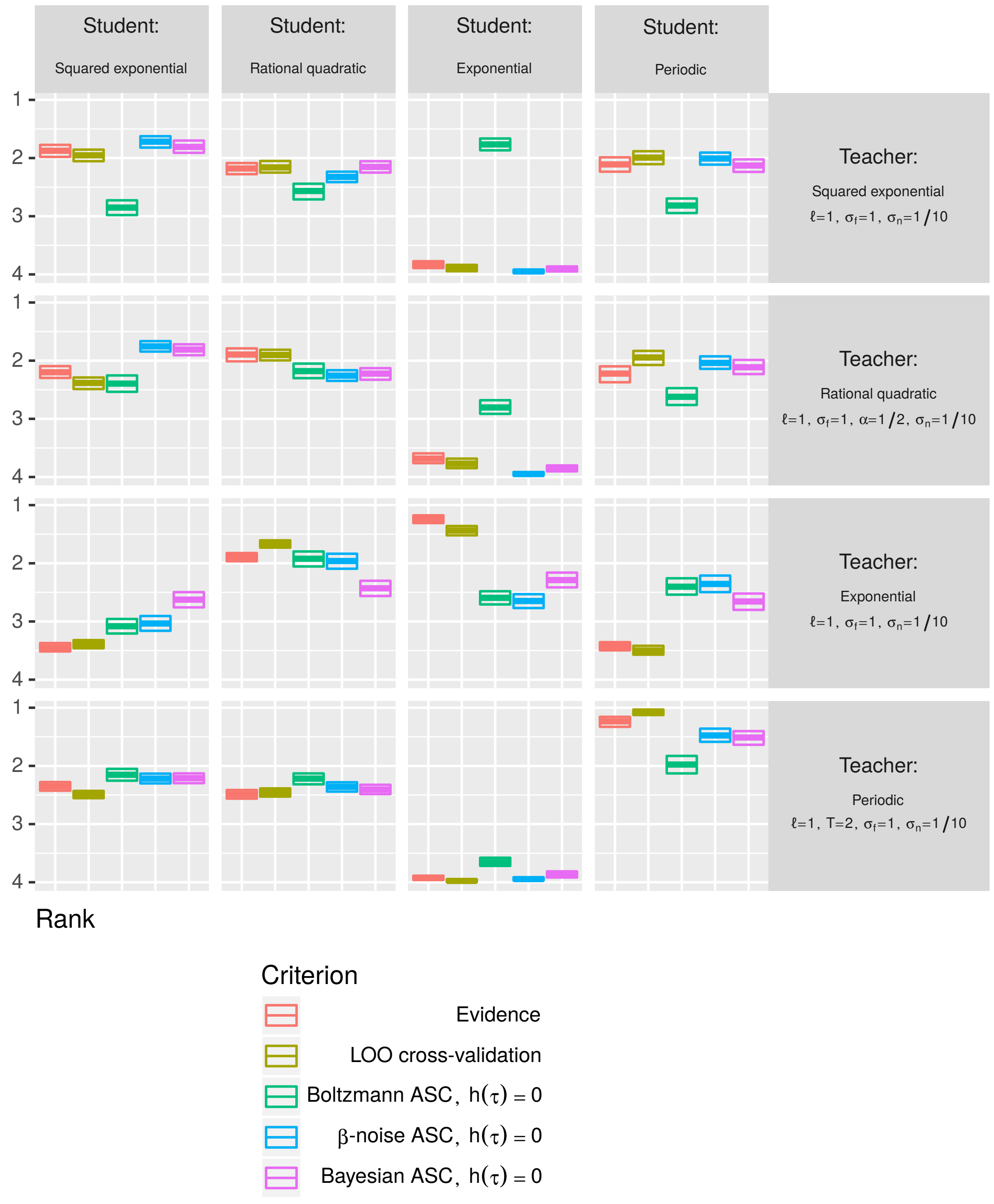


Figure 5.10.: Functional form selection for synthetic data drawn from the teacher model according to Experiment 6 on page 49. The hyperparameters are optimized by leave-one-out cross-validation for each student functional form. The resulting models are then ranked according to various criteria, with rank 1 being the best. The mean rank is visualized with a 95 % confidence interval.

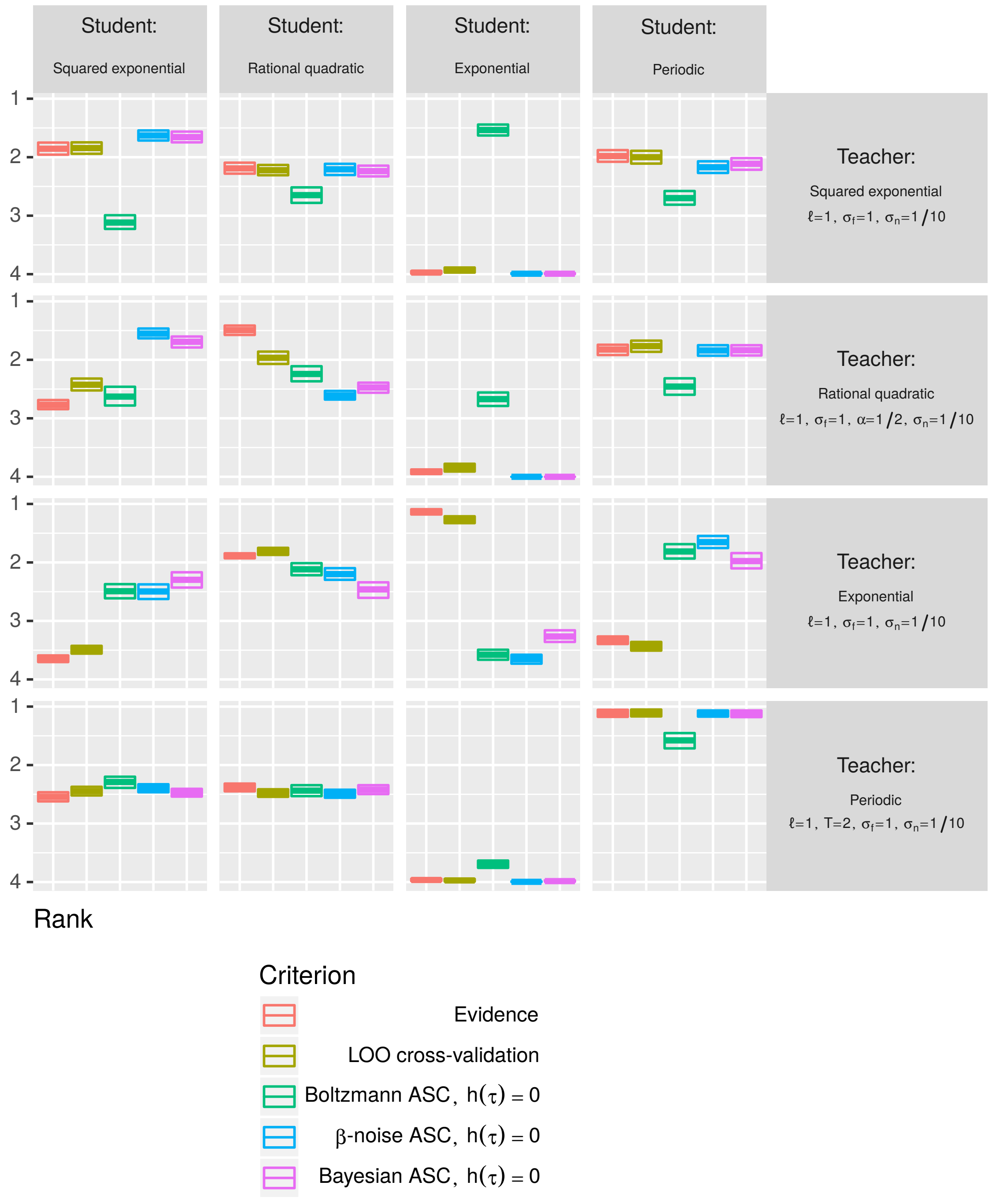


Figure 5.11.: Functional form selection for synthetic data drawn from the teacher model according to Experiment 7 on page 49. The hyperparameters are optimized by maximum evidence for each student functional form. The resulting models are then ranked according to various criteria, with rank 1 being the best. The mean rank is visualized with a 95 % confidence interval.

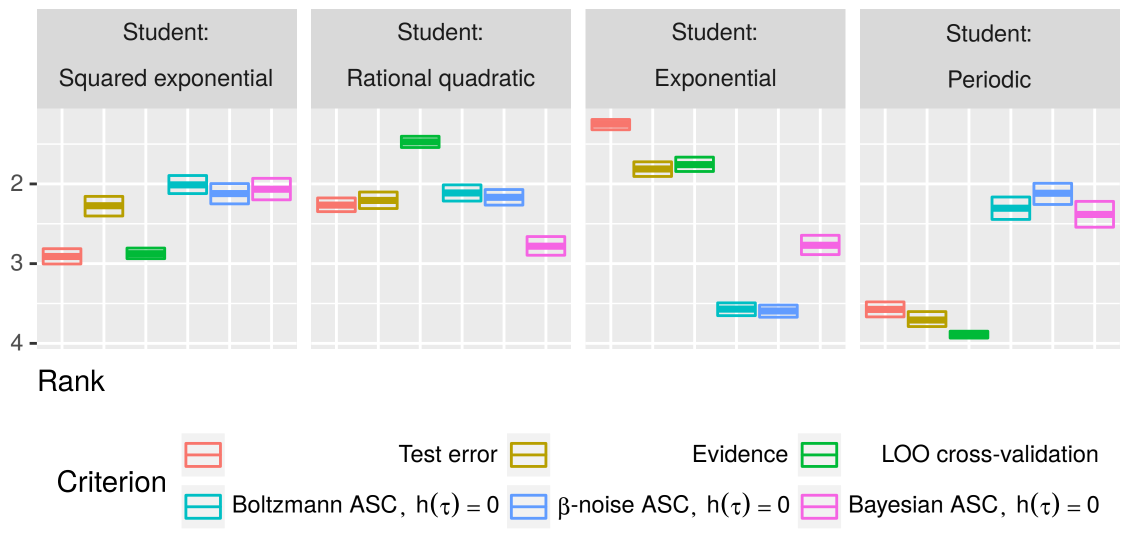


Figure 5.12.: Functional form selection for real-world data according to Experiment 8 on page 49. Given the training data, the hyperparameters are optimized by leave-one-out cross-validation to then rank them by various criteria, with rank 1 being the best. The test error on separate test data serves as a guide. The mean rank is visualized with a 95 % confidence interval.

number of student functional forms by leave-one-out cross-validation. We then rank the resulting models by various criteria, with rank 1 being the best. As a guide for comparison, we do the ranking according to the standardized mean squared error on the corresponding test data set. Approximation set coding is done with $J = 32$ partitions, $L = 2$ subsets, and $M = 8$ objects. See Figures 5.12 and 5.13 on this page and on the next page.

According to the test error, the exponential covariance function seems the most suitable on average, followed by the rational quadratic, squared exponential, and periodic covariance functions in that order. The criterion of maximum evidence exactly reproduces this ranking. Leave-one-out cross-validation favors the rational quadratic over the exponential covariance function, but otherwise suggests the same ranking. No approximation set coding with $h(\tau) = 0$ does well compared to the test error. On average, they consistently rank the exponential covariance function last, often favoring the squared exponential covariance function instead.

Figure 5.13 on the facing page illustrates the predictive means for an example partition. Note that the test errors for that partition have the same ranking as on average for many partitions. It can be seen that the squared exponential and the periodic covariance functions are steadier than the others, explaining the data with a higher noise level σ_n. The rational quadratic and especially the exponential covariance functions fit more

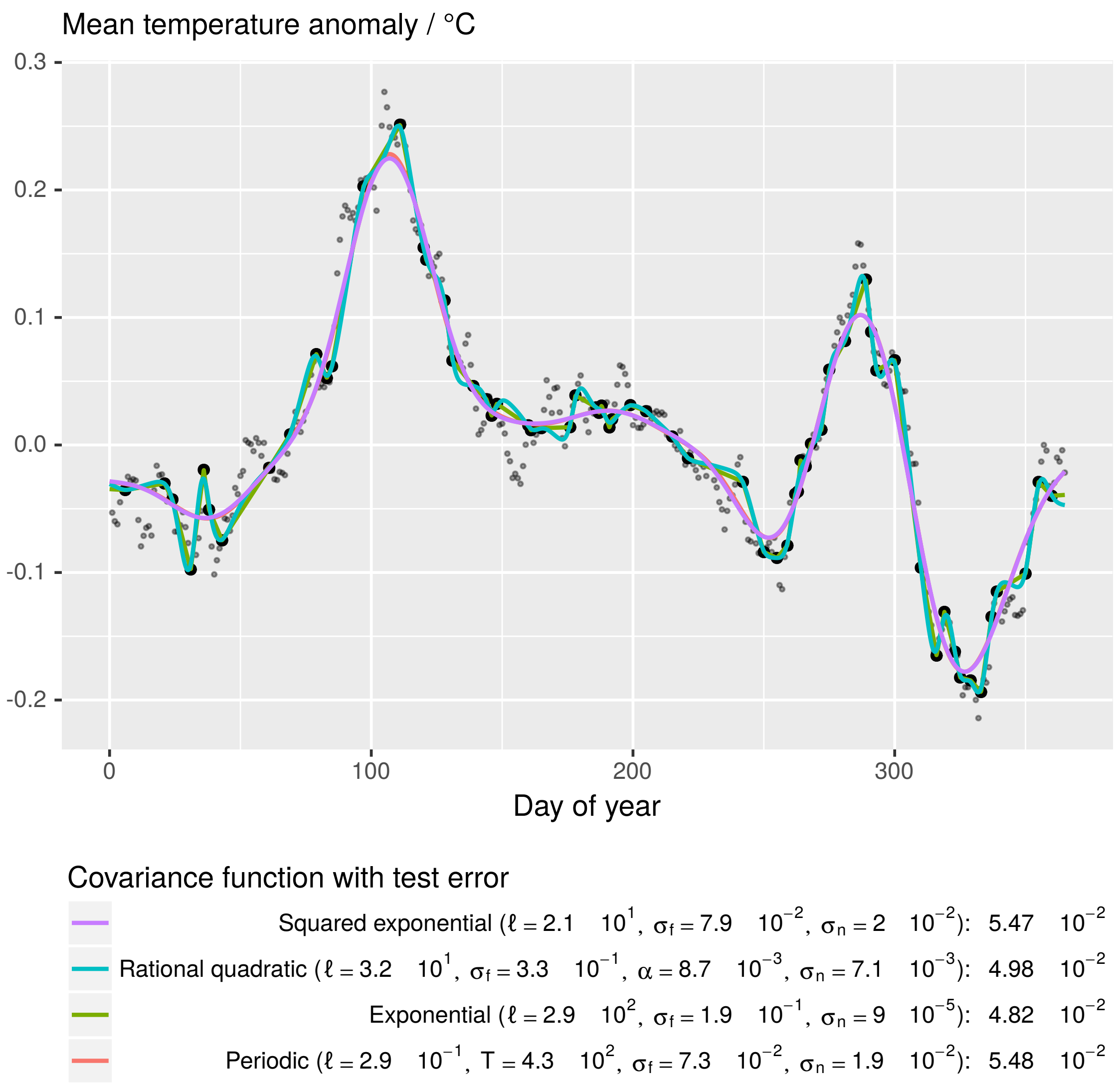


Figure 5.13.: Predictive mean for the real-world data according to Experiment 8 on page 49. The hyperparameters are optimized by leave-one-out cross-validation on an example partition into a training data set (big points) and a test data set (small points). The test error is the standardized mean squared error on the test data set. Note that mean of the squared exponential mostly overlaps the one of the periodic covariance function.

to the data, hence the lower noise level σ_n. Based on this manual evaluation, it seems not entirely sure which functional form should be best, as we face the central trade-off between underfitting and overfitting. Only focusing on the test error is perhaps overly simplistic. In the example, the periodic and the squared exponential covariance function both appear justified options, but the test error ranks them last, which is also what the classics of maximum evidence and leave-one-out cross-validation do on average.

5.4. Diversity of Approximation Set Coding

Not only does approximation set coding have several principal variants, but one has to decide for its diverse parameters. In particular, these are the number J of partitions, the number L of subsets, and the number M of objects. In the remainder, we explore these options, which at the same time explains the rationale behind typical setups.

Suppose that the M objects $\widetilde{\mathbf{X}} \in \mathbb{R}^{D \times M}$ are chosen uniformly at random from the inputs $\mathbf{X} \in \mathbb{R}^{D \times N}$ without replacement. Then the probability that a fixed case $\mathbf{x}_n$ of the data set is chosen as an object is M/N. Therefore, the chance that it is chosen at least once during J partitions equals

$$1 - \left(1 - \frac{M}{N}\right)^J . \tag{5.2}$$

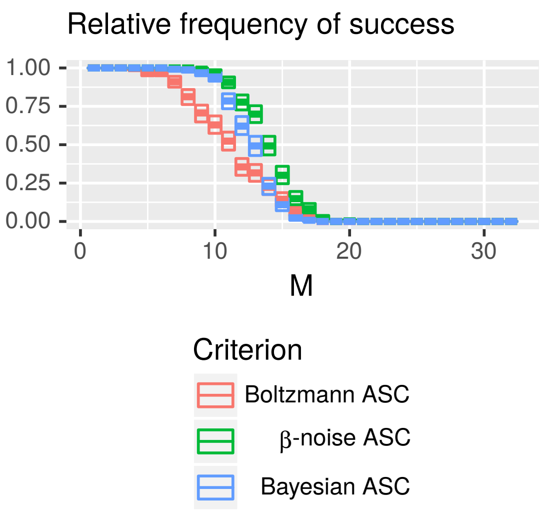
For a typical setup of $J = 32$, $M = 8$, and $N = 64$, this probability is approximately 0.99, so that the objects should cover the input distribution enough.

The decision for $L = 2$ subsets corresponds to the two-instance scenario in the original theory of approximation set coding. On the other hand, the number M of objects is an addition. The selection of M affects whether the objective function of approximation set coding can be numerically evaluated at all.

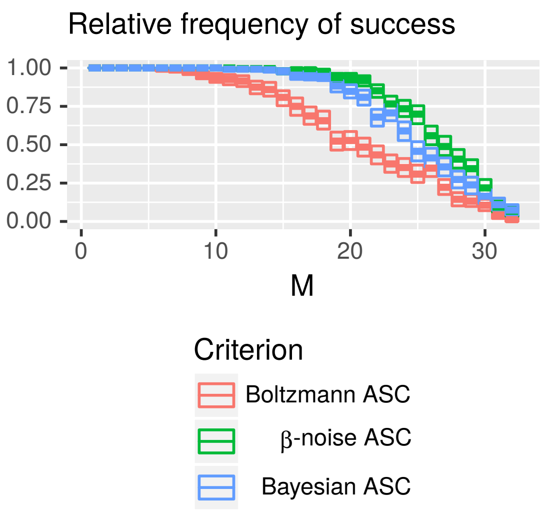
Experiment 9. We randomly draw 256 data sets from a Gaussian process, each of dimension $D = 1$ with $N = 64$ cases. Approximation set coding is done with $h(\tau) = 0$ and $L = 2$ subsets. For each data set, we try to compute the value of a criterion at the hyperparameters of the generative model. An evaluation of a criterion counts as a success if it is numerically possible to do the involved matrix decompositions and the optimization with respect to β, if any. The number of objects is varied in the range $M = 1, \ldots, 32$. See Figure 5.14 on the next page.

Except for Bayesian approximation set coding for the exponential covariance function, the chances for a success generally decrease with an increasing number M of objects. The decay is varies significantly with the covariance function, with the squared exponential covariance function being the most sensitive one. In conclusion, M needs to be adapted to the specific application. Instead of fixing M, an interesting twist could be to choose it at random.

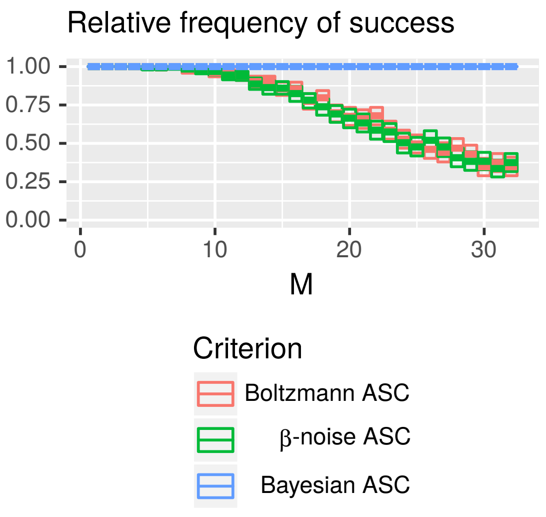
Next, we would like to explore how the mutual information I_β^θ of Boltzmann approximation set coding behaves as a function of the approximation precision β. In addition,



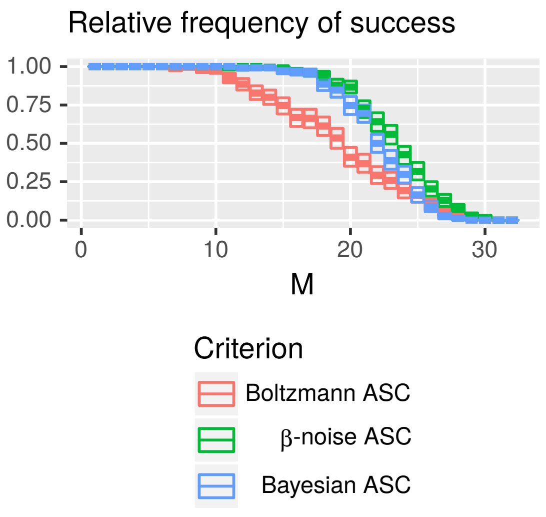
(a) Squared exponential covariance function where $\ell = 1$, $\sigma_f = 1$, and $\sigma_n = 1/10$.



(b) Rational quadratic covariance function where $\ell = 1$, $\sigma_f = 1$, $\alpha = 1/2$, and $\sigma_n = 1/10$.



(c) Exponential covariance function where $\ell = 1$, $\sigma_f = 1$, and $\sigma_n = 1/10$.



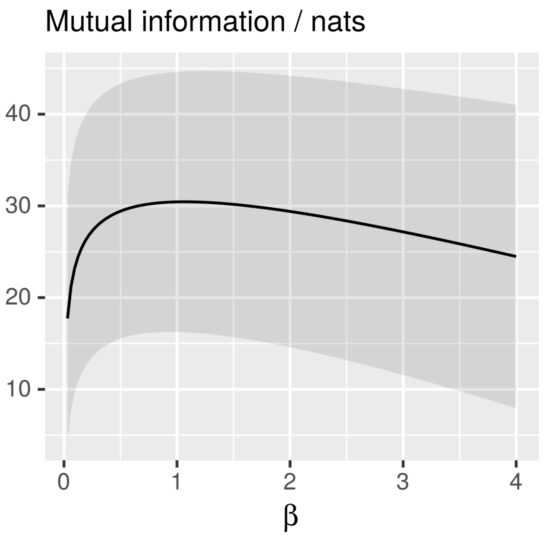
(d) Periodic covariance function where $\ell = 1$, $T = 2$, $\sigma_f = 1$, and $\sigma_n = 1/10$.

Figure 5.14.: Chances that approximation set coding can be numerically evaluated at all according to Experiment 9 on page 54. The mean is visualized with a 95 % confidence interval.

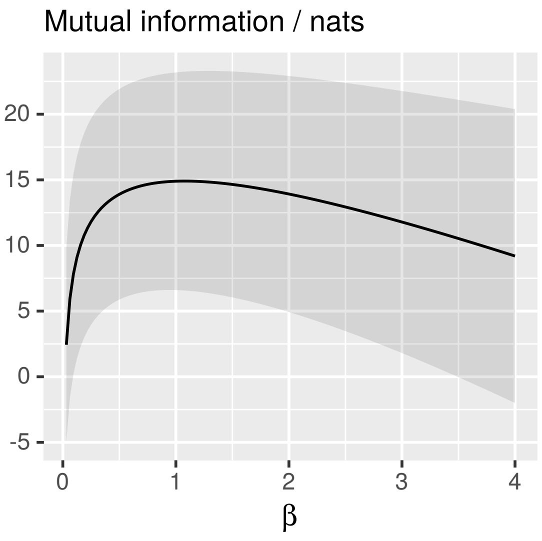
we thereby hope to gain insight into the connection to β-noise approximation set coding, which can be viewed as a special case fixing $\beta = 1$.

Experiment 10. Let a Gaussian process have hyperparameters θ. From it, we randomly draw 256 data sets, each of dimension $D = 1$ with $N = 64$ cases. Approximation set coding is done with $h(\tau) = 0$, $L = 2$ subsets, and $M = 8$ objects. For one drawn objective function per data set, we compute the mutual information I_β^θ as a function of the approximation precision β. The β-axis is discretized by 128 equally spaced points. See Figure 5.15 on the facing page.

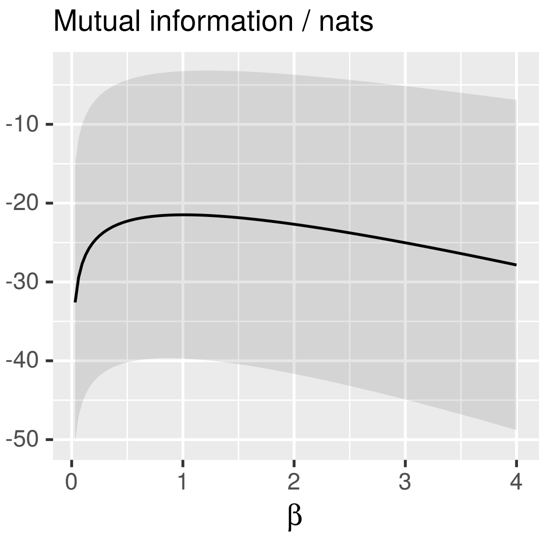
On average, the optimum seems to be close to $\beta = 1$, no matter what the covariance function is. Even though this could hint at a connection to β-noise approximation set coding, note that these objective functions are fixed to the hyperparameters θ of the generative model.



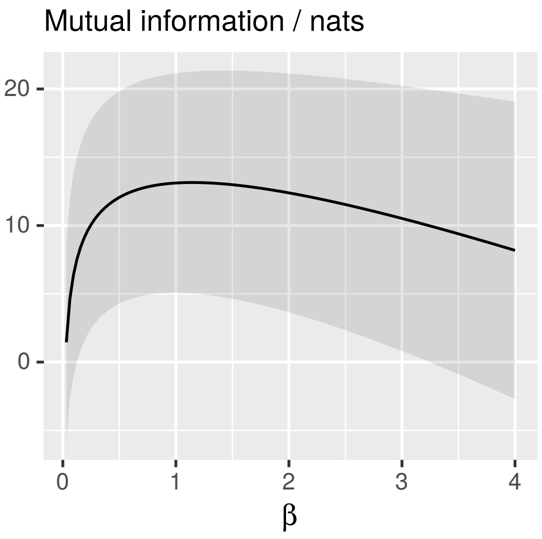
(a) Squared exponential covariance function where $\ell = 1$, $\sigma_f = 1$, and $\sigma_n = 1/10$.



(b) Rational quadratic covariance function where $\ell = 1$, $\sigma_f = 1$, $\alpha = 1/2$, and $\sigma_n = 1/10$.



(c) Exponential covariance function where $\ell = 1$, $\sigma_f = 1$, and $\sigma_n = 1/10$.



(d) Periodic covariance function where $\ell = 1$, $T = 2$, $\sigma_f = 1$, and $\sigma_n = 1/10$.

Figure 5.15.: Mean of mutual information I_β^ϑ of Boltzmann approximation set coding according to Experiment 10 on page 56.

6. Discussion

In light of our experiments, maximum evidence appears to be the best model selection criterion. It does remarkably well for both hyperparameter optimization and functional form selection. The other classic of leave-one-out cross-validation is a strong competitor, but typically suffers from more variance, which is a potential risk for model selection [8]. It would be interesting to see whether K-fold cross-validation for $K \neq N$ could significantly reduce this variance.

The performance of approximation set coding can be described as unsteady. Boltzmann and β-noise approximation set coding for $h(\tau) = 0$ seem to have the potential to approach leave-one-out cross-validation for hyperparameter optimization, but fall behind for functional form selection. Bayesian approximation set coding for $h(\tau) = 0$ seems to perform the other way around for synthetic data, often being able to identify the covariance function of the generative model. On the real-world data, approximation set coding is rather indecisive to rank the functional forms. One could interpret this uncertainty as a sign that no single functional form is really suitable, but further exploration is needed instead, considering other options such as combinations of covariance functions.

One should be careful not to overgeneralize about the performance of the criteria. The experiments of this work are very specific and by no means exhaustive, with the data sets being limited in their diversity. For example, the number N of cases is hardly ever altered. What further prevents us from drawing strong conclusions is the decision to settle for a single error measure. The standardized mean squared error of Equation (5.1) on page 39 solely takes the mean of the predictive distribution into account, but not its covariance. Whereas this error is applicable to any regression model, an alternative that assesses the full predictive distribution is the mean standardized log loss [30]. Moreover, note that the test error on the real-world data is assigned the critical responsibility of guiding which functional form to select. However, this is a quite simplistic way to tell which functional form suits best, as indicated by the visualized example predictions of Figure 5.13 on page 53.

Apart from the model selection abilities of the criteria, there are practical aspects summarized in Table 6.1 on the next page. Both maximum evidence and leave-one-out cross-validation are not hard to implement and run fast. There are no parameters to decide for, saving one from solving yet another selection problem. K-fold cross-validation would already pose the question of how to select the number K of folds. On the other end of the spectrum, approximation set coding is generally harder to implement and runs much slower. To give a rough idea of the scale, Boltzmann approximation set coding might

Criterion	Parameters	Complexity of implementation	Runtime
Evidence	None	Very simple	Fast
LOO cross-validation	None	Simple	Fast
Boltzmann ASC	Many	Complex	Slow
β-noise ASC	Many	Medium	Medium
Bayesian ASC	Many	Medium	Medium

Table 6.1.: Practical aspects of the criteria.

be implemented on 5 times as much lines of source code as maximum evidence, yet its runtime being 50 times longer. The parameters of approximation set coding are diverse. Not only do they include numbers such as the partition count J, but also strategies like choosing the objects $\widetilde{\mathbf{X}}$. Certainly, this can be overwhelming. At the same time, this flexibility offers a substantial potential for improvement still to be explored.

Our instantiation of approximation set coding comes with fundamental limitations that are hard to overcome so far. As seen earlier, the number M of objects cannot be that large, mainly for numerical reasons. One could try to add a diagonal matrix $\epsilon\mathbf{I}$ (for an insignificant ϵ) to numerically singular covariance matrices in order to make them invertible, at the danger of distorting the approximation capacity. Second, the expression $h(\tau)$ in approximation set coding is difficult to choose, and instantiating $h(\tau)$ differently could make a critical difference in terms of model selection. Lastly, the mutual information of approximation set coding is an error bound, which is potentially loose. In a similar vein with PAC-bounds [30], the optimum argument for the bound might be far from the optimum of the actual error.

7. Conclusion

We conclude with an outlook for future work. A possible cause for a restricted performance of approximation set coding could be the reduced number M of objects, since too many covariance matrices are numerically singular otherwise. To solve this problem, one could attempt to apply dependent Gaussian processes, which bypass the issue of singular covariance matrices by modeling multiple outputs for the same input [6]. A next step would be to apply approximation set coding to a more realistic scenario of model selection, in which mean functions and combinations of covariance functions are used to represent several characteristics of a data set.

As we have seen in Experiment 8 on page 49, the classics of maximum evidence and leave-one-out cross-validation can disagree on which covariance function to select for real-world data. Other criteria such as approximation set coding or K-fold cross-validation add yet more opinions about which model to select. It is thus natural to ask the question how one deals with this variety of criteria. Do we need a metacriterion to select a model selection criterion? Rather than picking a single criterion, one could try to combine them to make a decision that is stronger than each individual opinion. A simple way to combine them is to choose one criterion for hyperparameter optimization and a possibly different one for functional form selection – that is, to distinguish between R_2 and R_3 of Section 2.4.1 on page 18. Following the example of ensembles in decision making [28], it would be enlightening to see more work on combining model selection criteria.

A. Useful Derivatives

Proposition 13. *If $\mathbf{A}$ is invertible, then*

1. $$\frac{\partial}{\partial\gamma}\mathbf{A}^{-1} = -\mathbf{A}^{-1}\frac{\partial\mathbf{A}}{\partial\gamma}\mathbf{A}^{-1}$$

   [25, Equation (9)] and

2. $$\frac{\partial}{\partial\gamma}\log|\mathbf{A}| = \operatorname{tr}\left(\mathbf{A}^{-1}\frac{\partial\mathbf{A}}{\partial\gamma}\right)$$

   [19, Equation (0.8.10.1)].

Proposition 14.

$$\frac{\partial\log\mathcal{N}(\boldsymbol{\mu}\mid\mathbf{0},\boldsymbol{\Sigma})}{\partial\gamma} = -\frac{1}{2}\left(\boldsymbol{\mu}^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\left(2\frac{\partial\boldsymbol{\mu}}{\partial\gamma} - \frac{\partial\boldsymbol{\Sigma}}{\partial\gamma}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right) + \operatorname{tr}\left(\boldsymbol{\Sigma}^{-1}\frac{\partial\boldsymbol{\Sigma}}{\partial\gamma}\right)\right).$$

Proof. Using Proposition 1 on page 15, we have

$$\begin{aligned}\frac{\partial\log\mathcal{N}(\boldsymbol{\mu}\mid\mathbf{0},\boldsymbol{\Sigma})}{\partial\gamma} &= -\frac{1}{2}\left(\frac{\partial\boldsymbol{\mu}^{\mathsf{T}}}{\partial\gamma}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \boldsymbol{\mu}^{\mathsf{T}}\frac{\partial\boldsymbol{\Sigma}^{-1}}{\partial\gamma}\boldsymbol{\mu} + \boldsymbol{\mu}^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\frac{\partial\boldsymbol{\mu}}{\partial\gamma} + \frac{\partial\log|\boldsymbol{\Sigma}|}{\partial\gamma}\right)\\ &= -\frac{1}{2}\left(2\boldsymbol{\mu}^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\frac{\partial\boldsymbol{\mu}}{\partial\gamma} - \boldsymbol{\mu}^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\frac{\partial\boldsymbol{\Sigma}}{\partial\gamma}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \operatorname{tr}\left(\boldsymbol{\Sigma}^{-1}\frac{\partial\boldsymbol{\Sigma}}{\partial\gamma}\right)\right)\\ &= -\frac{1}{2}\left(\boldsymbol{\mu}^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\left(2\frac{\partial\boldsymbol{\mu}}{\partial\gamma} - \frac{\partial\boldsymbol{\Sigma}}{\partial\gamma}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right) + \operatorname{tr}\left(\boldsymbol{\Sigma}^{-1}\frac{\partial\boldsymbol{\Sigma}}{\partial\gamma}\right)\right).\end{aligned}$$

□

Proposition 15.

$$\frac{\partial\log\left(|\boldsymbol{\Lambda}|\mathcal{N}(\boldsymbol{\mu}\mid\mathbf{0},\boldsymbol{\Lambda})\right)}{\partial\gamma} = \frac{1}{2}\left(\boldsymbol{\mu}^{\mathsf{T}}\boldsymbol{\Lambda}^{-1}\left(\frac{\partial\boldsymbol{\Lambda}}{\partial\gamma}\boldsymbol{\Lambda}^{-1}\boldsymbol{\mu} - 2\frac{\partial\boldsymbol{\mu}}{\partial\gamma}\right) + \operatorname{tr}\left(\boldsymbol{\Lambda}^{-1}\frac{\partial\boldsymbol{\Lambda}}{\partial\gamma}\right)\right).$$

Proof. Using Proposition 14, we have

$$\frac{\partial\log\mathcal{N}(\boldsymbol{\mu}\mid\mathbf{0},\boldsymbol{\Lambda})}{\partial\gamma} = -\frac{1}{2}\left(\boldsymbol{\mu}^{\mathsf{T}}\boldsymbol{\Lambda}^{-1}\left(2\frac{\partial\boldsymbol{\mu}}{\partial\gamma} - \frac{\partial\boldsymbol{\Lambda}}{\partial\gamma}\boldsymbol{\Lambda}^{-1}\boldsymbol{\mu}\right) + \operatorname{tr}\left(\boldsymbol{\Lambda}^{-1}\frac{\partial\boldsymbol{\Lambda}}{\partial\gamma}\right)\right).$$

To that, we add

$$\frac{\partial\log|\boldsymbol{\Lambda}|}{\partial\gamma} = \operatorname{tr}\left(\boldsymbol{\Lambda}^{-1}\frac{\partial\boldsymbol{\Lambda}}{\partial\gamma}\right)$$

to arrive at the statement. □

Proposition 16. *If* $\boldsymbol{\mu} = \beta\mathbf{r} + \mathbf{s}$ *and* $\boldsymbol{\Lambda} = \beta\mathbf{P} + \mathbf{Q}$ *where* $\boldsymbol{\Lambda}$ *is symmetric positive-definite, then*

$$\frac{\partial \log\left(|\boldsymbol{\Lambda}|\,\mathcal{N}\left(\boldsymbol{\mu} \mid \mathbf{0}, \boldsymbol{\Lambda}\right)\right)}{\partial \log \beta} = \frac{\beta}{2}\left(\boldsymbol{\mu}^{\mathsf{T}}\boldsymbol{\Lambda}^{-1}\left(\mathbf{P}\boldsymbol{\Lambda}^{-1}\boldsymbol{\mu} - 2\mathbf{r}\right) + \operatorname{tr}\left(\boldsymbol{\Lambda}^{-1}\mathbf{P}\right)\right).$$

Proof. By the chain rule in calculus, we have

$$\frac{\partial \log\left(|\boldsymbol{\Lambda}|\,\mathcal{N}\left(\boldsymbol{\mu} \mid \mathbf{0}, \boldsymbol{\Lambda}\right)\right)}{\partial \log \beta} = \left(\frac{\partial \log \beta}{\partial \beta}\right)^{-1} \frac{\partial \log\left(|\boldsymbol{\Lambda}|\,\mathcal{N}\left(\boldsymbol{\mu} \mid \mathbf{0}, \boldsymbol{\Lambda}\right)\right)}{\partial \beta},$$

where we can then simply apply Proposition 15 on page 61. □

Bibliography

[1] Ed Anderson et al. *LAPACK Users' Guide*. 3rd edition. Society for Industrial and Applied Mathematics, 1999. ISBN: 978-0-89871-447-0 (cited on page 17).

[2] François Bachoc. "Cross Validation and Maximum Likelihood estimations of hyper-parameters of Gaussian processes with model misspecification". In: *Computational Statistics & Data Analysis* 66 (2013), pages 55–69. DOI: `10.1016/j.csda.2013.03.016` (cited on pages 8, 20, 49).

[3] *Berkeley Earth daily TAVG full dataset*. Berkeley Earth, January 23, 2015. URL: `http://www.berkeleyearth.org` (cited on pages 39, 40).

[4] Anil Kumar Bhattacharyya. "On a measure of divergence between two statistical populations defined by their probability distributions". In: *Bulletin of the Calcutta Mathematical Society* 35 (1943), pages 99–109. URL: `https://www.ams.org/mathscinet-getitem?mr=0010358` (cited on page 22).

[5] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. ISBN: 978-0-387-31073-2 (cited on page 15).

[6] Phillip Boyle and Marcus Frean. "Dependent Gaussian Processes". In: *Advances in Neural Information Processing Systems*. Volume 17. 2004, pages 217–224. URL: `http://papers.nips.cc/paper/2561-dependent-gaussian-processes` (cited on page 60).

[7] Joachim M. Buhmann. "SIMBAD: Emergence of Pattern Similarity". In: *Similarity-Based Pattern Analysis and Recognition*. Edited by Marcello Pelillo. Advances in Computer Vision and Pattern Recognition. Springer London, 2013, pages 45–64. ISBN: 978-1-4471-5627-7. DOI: `10.1007/978-1-4471-5628-4_3` (cited on pages 8, 20, 24).

[8] Gavin C. Cawley and Nicola L. C. Talbot. "On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation". In: *Journal of Machine Learning Research* 11 (2010), pages 2079–2107. ISSN: 1532-4435. URL: `http://www.jmlr.org/papers/v11/cawley10a.html` (cited on page 58).

[9] Olivier Chapelle. "Some thoughts about Gaussian Processes. Model Selection and Large Scale". 2005. URL: `http://is.tuebingen.mpg.de/fileadmin/user_upload/files/publications/gp_[0].pdf` (visited on 04/29/2016) (cited on page 20).

[10] Morteza Haghir Chehreghani, Alberto Giovanni Busetto, and Joachim M. Buhmann. "Information Theoretic Model Validation for Spectral Clustering". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. Volume 22. Journal of Machine Learning Research, 2012, pages 495–503. URL: `http://jmlr.org/proceedings/papers/v22/haghir12.html` (cited on pages 8, 20, 21, 24).

[11] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. 2nd edition. Wiley, 2006. ISBN: 978-0-471-24195-9 (cited on page 16).

[12] David Duvenaud. "Automatic Model Construction with Gaussian Processes". PhD thesis. University of Cambridge, 2014. URL: `http://people.seas.harvard.edu/~dduvenaud` (cited on page 18).

[13] David Duvenaud. *The Kernel Cookbook. Advice on Covariance functions*. 2015. URL: `http://people.seas.harvard.edu/~dduvenaud/cookbook/` (visited on 04/26/2016) (cited on pages 12, 39).

[14] Mario Frank and Joachim M. Buhmann. "Selecting the rank of truncated SVD by Maximum Approximation Capacity". In: *IEEE International Symposium on Information Theory*. IEEE, 2011, pages 1036–1040. arXiv: `1102.3176 [cs.IT]` (cited on pages 21, 22, 24–26).

[15] Isabelle Guyon et al. "Model Selection: Beyond the Bayesian/Frequentist Divide". In: *Journal of Machine Learning Research* 11 (2010), pages 61–87. URL: `http://www.jmlr.org/papers/v11/guyon10a.html` (cited on page 18).

[16] David Hall and Daniel Ramage. *Breeze*. Version 0.12. URL: `https://github.com/scalanlp/breeze` (cited on page 38).

[17] Sam Halliday. *netlib-java*. Version 1.1.2. URL: `https://github.com/fommil/netlib-java` (cited on page 38).

[18] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. 2nd edition. Society for Industrial and Applied Mathematics, 2002. ISBN: 978-0-89871-521-7 (cited on page 17).

[19] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. 2nd edition. Cambridge University Press, 2012. ISBN: 978-0-521-83940-2 (cited on pages 16, 17, 61).

[20] Joshua Kaplan. *matlabcontrol*. Version 4.1.0. URL: `https://code.google.com/archive/p/matlabcontrol/` (cited on page 38).

[21] Ron Kohavi. "A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection". In: *International Joint Conference on Artificial Intelligence*. Volume 2. Morgan Kaufmann, 1995, pages 1137–1143. URL: `http://dl.acm.org/citation.cfm?id=1643047` (cited on page 20).

[22] Stuart P. Lloyd. "Least Squares Quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2 (1982), pages 129–137. DOI: `10.1109/TIT.1982.1056489` (cited on page 20).

[23] Craig Lucas. *LAPACK-Style Codes for Level 2 and 3 Pivoted Cholesky Factorizations*. LAPACK Working Note 161. University of Manchester, 2004. URL: http://www.netlib.org/lapack/lawnspdf/lawn161.pdf (cited on page 17).

[24] David J. C. MacKay. "Introduction to Gaussian Processes". In: *NATO ASI Series F Computer and Systems Sciences* 168 (1998), pages 133–166. URL: http://www.inference.eng.cam.ac.uk/mackay/abstracts/gpB.html (cited on page 14).

[25] Thomas P. Minka. *Old and New Matrix Algebra Useful for Statistics*. Technical report. MIT Media Lab, 2000. URL: http://research.microsoft.com/en-us/um/people/minka/papers/matrix/ (cited on page 61).

[26] Radford M. Neal. *Bayesian Learning for Neural Networks*. Lecture Notes in Statistics. Springer New York, 1996. ISBN: 978-0-387-94724-2 (cited on page 14).

[27] Jorge Nocedal. "Updating Quasi-Newton Matrices With Limited Storage". In: *Mathematics of Computation* 35.151 (1980), pages 773–782. DOI: 10.1090/S0025-5718-1980-0572855-7 (cited on pages 6, 18, 38, 40).

[28] Robi Polikar. "Ensemble Based Systems in Decision Making". In: *IEEE Circuits and Systems Magazine* 6.3 (2006), pages 21–45. DOI: 10.1109/MCAS.2006.1688199 (cited on page 60).

[29] Carl Edward Rasmussen and Hannes Nickisch. *GPML Toolbox*. Version 3.6. URL: http://www.gaussianprocess.org/gpml/code/matlab/doc/ (cited on page 38).

[30] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. The MIT Press, 2006. ISBN: 978-0-262-18253-9 (cited on pages 8, 11, 14, 18, 19, 37, 58, 59).

[31] Matthias Seeger. "PAC-Bayesian Generalisation Error Bounds for Gaussian Process Classification". In: *Journal of Machine Learning Research* 3 (2002), pages 233–269. URL: http://www.jmlr.org/papers/v3/seeger02a.html (cited on page 19).

[32] Anders Skajaa. "Limited Memory BFGS for Nonsmooth Optimization". Master's thesis. New York University, 2010. URL: http://cs.nyu.edu/overton/mstheses/skajaa/msthesis.pdf (cited on page 18).

[33] Yung Liang Tong. *The Multivariate Normal Distribution*. Springer, 1990. ISBN: 978-1-4613-9657-4 (cited on pages 14, 15).

[34] George E. Uhlenbeck and Leonard S. Ornstein. "On the Theory of the Brownian Motion". In: *Physical Review* 36.5 (1930), pages 823–841. DOI: 10.1103/PhysRev.36.823 (cited on page 14).

[35] Hadley Wickham and Winston Chang. *ggplot2*. Version 2.1.0. URL: http://ggplot2.org (cited on page 40).

[36] Anthony Zee. *Quantum Field Theory in a Nutshell*. Princeton University Press, 2003. ISBN: 978-0-691-01019-9 (cited on page 16).

[37] Xiangxin Zhu et al. “Predicting Simulation Parameters of Biological Systems Using a Gaussian Process Model”. In: *Statistical Analysis and Data Mining* 5.6 (2012), pages 509–522. DOI: `10.1002/sam.11163` (cited on page 8).

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Model Selection for Gaussian Process Regression by Approximation Set Coding

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Fischer	Benjamin

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zurich, May 9, 2016	B. Fischer

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.