# Bayesian Kernel Methods[⋆]

Alexander J. Smola[1] and Bernhard Schölkopf[2]

[1] RSISE, The Australian National University, Canberra 0200, ACT, Australia
[2] Max Planck Institut für Biologische Kybernetik, 72076 Tübingen, Germany

**Abstract.** Bayesian methods allow for a simple and intuitive representation of the function spaces used by kernel methods. This chapter describes the basic principles of Gaussian Processes, their implementation and their connection to other kernel-based Bayesian estimation methods, such as the Relevance Vector Machine.

## 1 Introduction

The Bayesian approach allows for an intuitive incorporation of prior knowledge into the process of estimation. Moreover, it is possible, within the Bayesian framework, to obtain estimates of the confidence and reliability of the estimation process itself. These estimates are typically relatively easy to compute.

Surprisingly enough, as we shall see, the Bayesian approach leads to algorithms much akin to those developed within the framework of risk minimization. This allows us to provide new insight into kernel algorithms such as SV classification and regression. In addition, these similarities help us design Bayesian counterparts for risk minimization algorithms (such as Laplacian Processes (Section 5)), or vice versa (Section 6). In other words, we can tap into the knowledge from both worlds and combine it to create better algorithms.

We begin in Section 2 with an overview of the basic assumptions underlying Bayesian estimation. We explain the notion of prior distributions, which encode our prior belief concerning the likelihood of obtaining a certain estimate, and the concept of the posterior probability, which quantifies how plausible functions appear after we observe some data. Furthermore we show how inference is performed, and how certain numerical problems that arise can be alleviated by various types of Maximum-a-Posteriori (MAP) estimation.

Once the basic tools are introduced, we analyze the specific properties of Bayesian estimators for three different types of prior probabilities: Gaussian Processes (Section 3 describes the theory and Section 4 the implementation), which rely on the assumption that adjacent coefficients are correlated, Laplacian Processes (Section 5), which assume that estimates can be expanded into a sparse linear combination of kernel functions, and therefore favor such hypotheses, and Relevance Vector Machines (Section 6), which assume that the contribution of each kernel function is governed by a normal distribution with its own variance.

[⋆] The present article is based on [62].

Readers interested in a quick overview of the principles underlying Bayesian statistics will find the introduction sufficient. We recommend that the reader focus first on Sections 2 and 3. The subsequent sections are ordered in increasing technical difficulty, and decreasing bearing on the core issues of Bayesian estimation with kernels.

## 2 Bayesics

The central characteristic of Bayesian estimation is that we assume certain prior knowledge or beliefs about the data generating process, and the dependencies we might encounter. It is a natural extension of the maximum likelihood framework (loosely speaking the prior knowledge behaves exactly in the same way as additional data that we might have observed prior to the actual estimation problem).

Unless stated otherwise, we observe an $m$-sample $X := \{x_1, \ldots, x_m\}$ and $Y := \{y_1, \ldots, y_m\}$, based on which we will carry out inference, typically on a set of observations $X' := \{x'_1, \ldots, x'_{m'}\}$. For notational convenience we sometimes use $Z := \{(x_1, y_1), \ldots, (x_m, y_m)\}$ instead of $X, Y$. We begin with an overview over the fundamental ideas (see also [3, 40, 46, 63, 56, 62] for more details).[1]

### 2.1 Maximum Likelihood and Bayes Rule

Assume that we are given a set of observations $Y$ which are drawn from a probability distribution $p_\theta(Y)$. It then follows that for a given value of $\theta$ we can assess how likely it is to observe $Y$. Conversely, given the observations $Y$, we can try to find a value of $\theta$ for which $Y$ was a particularly likely observation.

For instance, if we know that $Y$ is composed of several observations of a scalar random variable drawn from a normal distribution with mean $\mu$ and variance $\sigma$, that is $\theta = (\mu, \sigma)$, we could try to infer $\mu$ and $\sigma$ from $Y$ or vice versa, assess how likely it is that $Y$ came from such a normal distribution.

The process of determining $\theta$ from data by maximizing $p_\theta(Y)$ is referred to as maximum likelihood estimation. We obtain

$$\theta_{\mathrm{ML}}(Y) := \underset{\theta}{\operatorname{argmax}}\, p_\theta(Y). \tag{1}$$

Note that we deliberately write $p_\theta(Y)$ instead of the conditional probability distribution $p(Y|\theta)$, since the latter implies that also $\theta$ is a random variable with a certain probability distribution $p(\theta)$, an additional assumption we may not want to make. Note that $p_\theta(Y)$, viewed as a function of $\theta$, is often referred to as the *likelihood* of $Y$, given $\theta$.

[1] For the sake of simplicity we will gloss over details such as $\sigma$-algebras. With some abuse of notation, $p(x)$ will correspond to either a density or a probability measure, depending on the context.

If, however, such a $p(\theta)$ exists, we can use the definition of conditional distributions

$$p(Y,\theta) = p(Y|\theta)p(\theta) = p(\theta|Y)p(Y) \tag{2}$$

and obtain Bayes' rule

$$p(\theta|Y) = \frac{p(Y|\theta)p(\theta)}{p(Y)} \propto p(Y|\theta)p(\theta). \tag{3}$$

This means that as soon as we assume that $\theta$ itself is a random variable with $p(\theta)$, we can infer how likely it is to observe a certain value of $\theta$, given $Y$. In this way, $\theta$ is transformed from the parameter of a density to a random variable, which allows us to make statements about the likely value of $\theta$ in the presence of data.

Typically $p(\theta|Y)$ is referred to as the *posterior* distribution, whereas $p(\theta)$ is known as the *prior*, and $p(Y|\theta)$ is called the *evidence*: with a given prior, based on our prior knowledge about $\theta$ and using the evidence of the observations $Y$ we can come to a posterior assessment $p(\theta|Y)$ on the probability of having observed a certain value of $\theta$.

Finally, the mode of $p(\theta|Y)$ is used in maximum a posteriori (MAP) estimation to find a good estimate for $\theta$ via

$$\theta_{\mathrm{MAP}}(Y) := \operatorname*{argmax}_{\theta}\, p(Y|\theta)p(\theta). \tag{4}$$

Note that the use of $\theta_{\mathrm{MAP}}(Y)$ is typically preferred to $\theta_{\mathrm{ML}}(Y)$, since we can avoid "unreasonable" values of $\theta$ by making suitable prior assumptions on $\theta$, such as its range.

## 2.2 Examples of Maximum Likelihood Estimation

In the following, we will be introducing various models of $p_\theta(y)$, which will become useful in regression and classification problems at a later stage in this chapter. We begin with the (simplifying) assumption that $Y$ is obtained iid (identically independently distributed) from $p_\theta(y)$, that is

$$p_\theta(Y) = \prod_{i=1}^{m} p_\theta(y_i). \tag{5}$$

and that furthermore $p_\theta(y) = p(y-\theta)$, where $p$ is a distribution with zero mean. In other words, we begin by studying the "estimation of a location parameter problem". Depending on the properties of $p(y-\theta)$ we will obtain various solutions for $p_{\mathrm{ML}}(Y)$.

For the practical purpose of finding $\theta_{\mathrm{ML}}(Y)$ under the assumption (5) it is advantageous to rewrite (1) as

$$\theta_{\mathrm{ML}}(Y) = \operatorname*{argmin}_{\theta} \sum_{i=1}^{m} -\log p(y_i - \theta). \tag{6}$$

This reduces the problem of maximizing a joint function of all $y_i$ to minimizing the average of terms, each of which is dependent only on one of the instances $y_i$. The term $-\log p(y-\theta)$, considered as a function of $\theta$ is often referred to as the *negative log-likelihood.*

**Normal Distribution:** we assume that we have a normal distribution with fixed variance $\sigma > 0$ and zero mean, that is

$$p(\xi) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}\xi^2\right) \tag{7}$$

and consequently $-\log p(y-\theta) = \frac{1}{2\sigma^2}(y-\theta)^2 + c$, where $c$ is a constant independent of $y$ and $\theta$. This means that $\theta_{\mathrm{MP}}(Y)$ satisfies

$$\theta_{\mathrm{MP}}(Y) = \operatorname*{argmin}_{\theta} \sum_{i=1}^{m} \frac{1}{2\sigma^2}(y_i - \theta)^2. \tag{8}$$

Taking derivatives, simple algebra shows that in this case

$$\theta_{\mathrm{MP}}(Y) = \frac{1}{m}\sum_{i=1}^{m} y_i. \tag{9}$$

In other words, $\theta_{\mathrm{MP}}(Y)$ for the normal distribution leads to the mean of random variables to estimate the location parameter.

**Laplacian Distribution:** again this distribution has zero mean, yet much longer tails than the normal distribution. It satisfies

$$p(\xi) = \frac{1}{2\lambda} \exp\left(-\frac{1}{\lambda}|\xi|\right) \tag{10}$$

and consequently $-\log p(y-\theta) = \frac{1}{\lambda}|y-\theta| + c$, where $c$ is a constant independent of $y$ and $\theta$. This means that $\theta_{\mathrm{MP}}(Y)$ satisfies

$$\theta_{\mathrm{MP}}(Y) = \operatorname*{argmin}_{\theta} \sum_{i=1}^{m} \frac{1}{2\lambda}|y_i - \theta|. \tag{11}$$

Taking derivatives, we can see that for the minimizer $\theta_{\mathrm{MP}}$ as many terms must satisfy $y_i > \theta$ as there are terms with $y_i < \theta$. Consequently, the median of $Y$ is the solution for $\theta_{\mathrm{MP}}$.

Clearly the median is a much more robust way of estimating the expected value of a distribution:[2] if we corrupt $Y$ with some additional data not taken from the true distribution we will obtain a good estimate nonetheless, regardless of the (possibly large) numerical value of the corrupted additional observations. This is the case since we excluded the extreme values in $Y$ on both ends.

[2] We assume that all distributions we are dealing with are symmetric and have zero mean, hence mean and median coincide.

**Robust Estimation:** Using only the median of $Y$ for estimation of $\theta$ appears to be too drastic, especially if we have reason to believe that not all the data is corrupted. Instead, Huber [29] formalized the notion of using a *trimmed mean*, where one only discards a certain fraction of extreme values and takes the mean of the rest. For this purpose the following density was introduced:

$$p(\xi) \propto \begin{cases} \exp(-\frac{\xi^2}{2\sigma}) & \text{if } |\xi| \leq \sigma \\ \exp(\frac{\sigma}{2} - |\xi|) & \text{otherwise} \end{cases} \tag{12}$$

One can show that $\theta_{\mathrm{ML}}(Y)$ is given by the mean of the fraction of observations that lie within an interval of $\pm\sigma$ around $\theta_{\mathrm{ML}}(Y)$ and where equal amounts of observations $y_i$ exceed $\theta_{\mathrm{ML}}(Y)$ by more than $\sigma$ from above and below.

**$\varepsilon$-insensitive Density:** For computational convenience Vapnik [75] introduced another variant of density model, based on the $\varepsilon$-insensitive loss function. It is essentially a Laplacian distribution, where in a neighborhood of size $\varepsilon$ around its mean all data is equally probable. We can formalize this as follows:

$$p(\xi) = \frac{1}{2(1+\varepsilon)} \exp(-|\xi|_\varepsilon) \text{ where } |\xi|_\varepsilon := \max(0, |\xi| - \varepsilon). \tag{13}$$

Estimators of $\theta_{\mathrm{ML}}(Y)$ have similar properties to the ones in the previous paragraph, with the difference that one takes the mean of the two extreme values in the $\varepsilon$-neighborhood of the expectation rather than the mean over the whole set. The advantage of this somewhat peculiar estimator is that optimization problems arising from it have a lower number of active constraints. This was exploited in Support Vector regression [75, 76].

Besides estimating the mean of a real-valued random variable $y$ we may also have to deal with discrete valued ones. For simplicity we consider only binary $y$, that is $y \in \{\pm 1\}$. In this case it is most useful to estimate the probability $p(y=1)$ (and $p(y=-1) = 1 - p(y=1)$).[3] While we could do this directly, it pays to consider a few indirect strategies for finding such an estimate. The reason is that at a later stage we will use this indirect parameterization to estimate $p(y=1)$ as a function of some additional parameter $x$ (in which case we will call $y$ the label and $x$ the pattern), a process which is greatly simplified by an indirect parameterization.

Typically we will study probabilities parameterized by $p_\theta(y) = p(\theta y)$, where $p$ may take on various functional forms. Often in classification, when $\theta$ is location dependent, the values $\theta(x)y$ are referred to as the *margin* at location $x$.

**Logistic Transfer Function:** this is given by

$$p_\theta(y) := \frac{\exp(\theta y)}{1 + \exp(\theta y)}. \tag{14}$$

[3] As in the real-valued case, we develop the reasoning for $p(y)$ rather than $p(y|x)$ and will introduce the conditioning part later in Section 3.

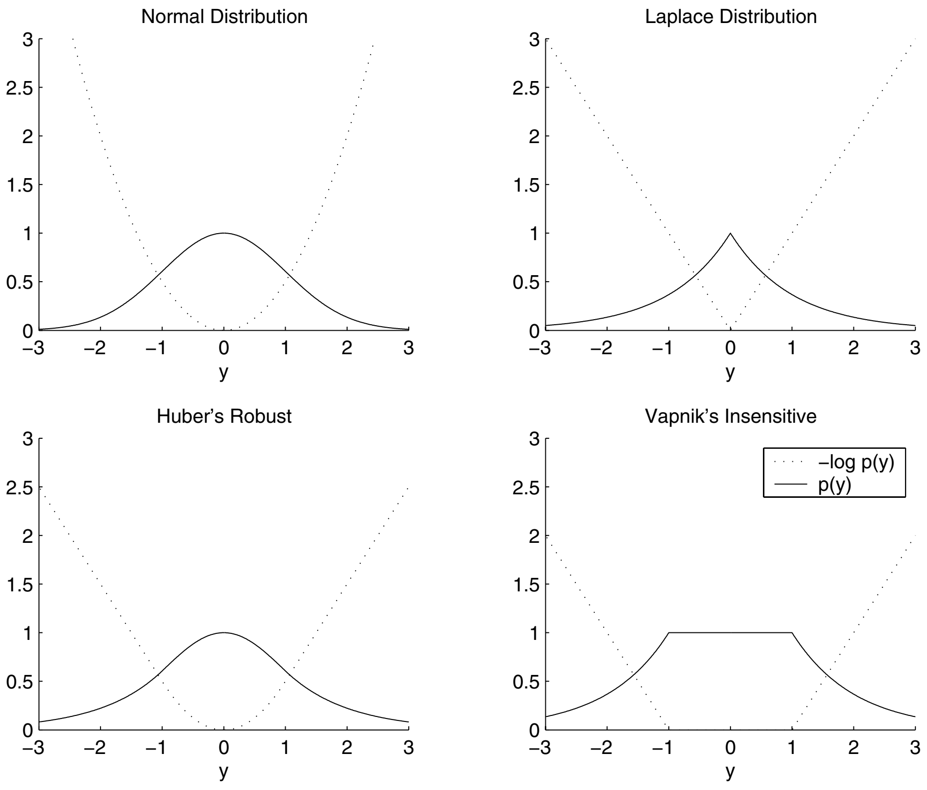

**Fig. 1.** Densities and corresponding negative log density. Upper left: Gaussian, upper right: Laplacian, lower left: Huber's robust, lower right: $\varepsilon$-insensitive.

In other words, $p(y = 1) = \frac{e^\theta}{1+e^\theta}$ and $p(y = -1) = \frac{e^{-\theta}}{1+e^{-\theta}}$. Note that logistic regression with (14) is equivalent to obtaining $\theta$ via

$$\theta = \ln \frac{p(y = 1)}{p(y = -1)}. \tag{15}$$

Quite often the logarithm of the ratio between the two class probabilities is also referred to as the *log odds ratio.*

**Probit:** we might also assume that $y$ is given by the sign of $\theta$, but corrupted by Gaussian noise (see for instance [49, 50, 63]); thus, $y = \operatorname{sgn}(\theta + \xi)$ where $\xi \sim \mathcal{N}(0, 1)$. In this case, we have

$$p_\theta(y) = \int \frac{\operatorname{sgn}(\theta + \xi) + 1}{2} p(\xi) d\xi \tag{16}$$

$$= \frac{1}{\sqrt{2\pi}} \int_{-\theta}^{\infty} \exp\left(-\frac{1}{2}\xi^2\right) d\xi = \Phi(\theta). \tag{17}$$

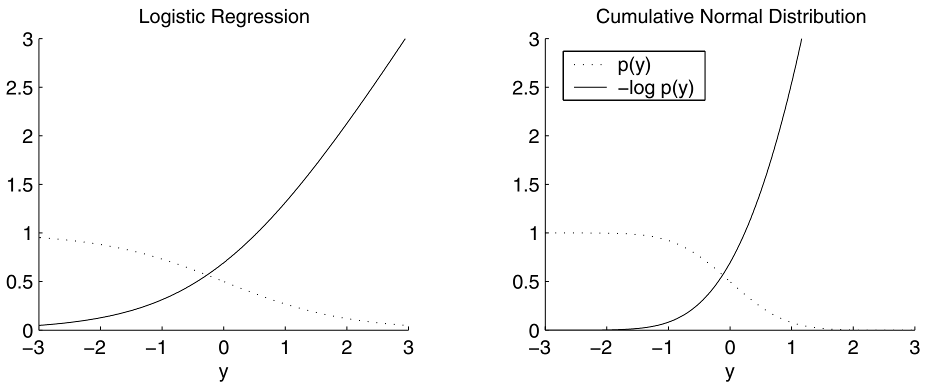

**Fig. 2.** Conditional probabilities and corresponding negative log-probability. Left: logistic regression; Right: Probit.

Here $\Phi$ is the distribution function of the normal distribution.

Note the correspondence between the asymptotic behaviors of the logistic and the probit model on the one hand, and the linear and quadratic soft margin loss functions $(1-\theta y)_+$ and $(1-\theta y)_+^2$. This also explains why the linear soft margin loss is a good proxy to logistic regression and the quadratic soft margin to the probit model. Furthermore, it indicates, that should one use the *linear* soft margin as a cheap proxy for optimization purposes, the logistic is a more adequate model to fit the densities subsequently [51], whereas, for the *quadratic* soft margin the probit model is to be preferred.

## 2.3 Inference

Besides the problem of finding suitable estimates of $\theta$ for $p_\theta(Y)$, which can be used to *understand* the way observations $Y$ were generated from some underlying distribution, we may also simply want to infer new values $Y'$, given some observations $Y$. For this purpose we need to compute $p(Y'|Y)$. The factorization of conditional probabilities leads to

$$p(Y'|Y) = \frac{p(Y',Y)}{p(Y)} \propto p(Y',Y). \tag{18}$$

This means that as soon as we can compute the *joint* probability distribution function of $Y, Y'$ we are able to predict $Y'$, given $Y$. The normalization term $p(Y)$ is not always needed — for instance, the mode of $p(Y'|Y)$ is independent of the scale.

*Example 1 (Inference with Normal Distributions).* Assume that $(Y, Y')$ is jointly normal with covariance matrix $\Sigma = \begin{bmatrix} \Sigma_{YY} & \Sigma_{YY'} \\ \Sigma_{Y'Y} & \Sigma_{Y'Y'} \end{bmatrix}$ and mean $\mu = \begin{bmatrix} \mu_Y \\ \mu_{Y'} \end{bmatrix}$, then $p(Y'|Y)$ is drawn from a normal distribution with covariance $\Sigma_{\mathrm{cond}} = \Sigma_{Y'Y'} - \Sigma_{Y'Y}\Sigma_{YY}^{-1}\Sigma_{YY'}$ and mean $\mu_{\mathrm{cond}} = \mu_{Y'} + \Sigma_{Y'Y}\Sigma_{YY}^{-1}(Y - \mu_Y)$.

This can be seen as follows: since $p(Y, Y')$ is jointly normal, the conditional distribution $p(Y'|Y)$ is also a normal distribution, which can be obtained from $p(Y, Y')$ by collecting all the terms which depend on $Y'$. We know that $p(Y, Y')$ is given by

$$(2\pi)^{-\frac{m+m'}{2}}(\det \Sigma)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\begin{bmatrix} Y - \mu_Y \\ Y' - \mu_{Y'} \end{bmatrix}^\top \begin{bmatrix} \Sigma_{YY} & \Sigma_{YY'} \\ \Sigma_{Y'Y} & \Sigma_{Y'Y'} \end{bmatrix}^{-1} \begin{bmatrix} Y - \mu_Y \\ Y' - \mu_{Y'} \end{bmatrix}\right)$$

Writing out the inverse of $\Sigma$ (see e.g., [39]) and collecting terms yields the above result.

In the next section we will use the above example to perform Gaussian Process prediction. For the moment just note that once we know $p(Y, Y')$ it is very easy to estimate $Y'$, given $Y$, since $p(Y'|Y) \propto p(Y, Y')$.

Quite often, unfortunately, we will not have $p(Y, Y')$ at our disposition immediately. Instead, we may only have $p(Y, Y'|\theta)$ together with $p(\theta)$. For instance, in all settings described in Section 2.2 we assumed to know the distribution of the observations only up to their expected value. This means that in order to obtain $p(Y, Y')$ we need to integrate out the *latent variable* $\theta$. This is achieved as follows:

$$p(Y, Y') = \int p(Y, Y', \theta)d\theta = \int p(Y, Y'|\theta)p(\theta)d\theta. \tag{19}$$

Eq. (19) may or may not be computable in closed form. Hence there exist various strategies to deal with the problem of obtaining $p(Y, Y')$. We list some of them below.

**Exact Solution:** If we can solve for $p(Y, Y')$ explicitly we can proceed as before with our estimation procedure after solving the integral. An important special case is where $(Y - \theta, Y' - \theta') \sim \mathcal{N}(\mu, \Sigma)$ and furthermore $(\theta, \theta') \sim \mathcal{N}(0, \Lambda)$, where $\theta, Y \in \mathbb{R}^m$, $\theta', Y' \in \mathbb{R}^{m'}$, and $\Lambda \in \mathbb{R}^{(m+m')^2}$.
Such a situation may occur where $Y$ (and $Y'$) are composed of two random variables, each of them normally distributed with their own mean and covariance. Typically $\theta$ assumes the role of the additive noise and $\Lambda$ is a multiple of the unit matrix. This, however need not be the case: we could deal with colored noise as well.
By construction $Y$, too, is normally distributed, where we simply add up the means and variances of the two constituents to obtain $(Y, Y') \sim \mathcal{N}(\mu, \Sigma + \Lambda)$. Hence, without much effort, we computed integral (19) by remarking that for normal distributions mean and variance add up. Note that inference in this situation proceeds identically to the discussion of Example 1). We will later in this chapter use this setting under the name 'Gaussian Process Regression with Normal Noise'.

**Sampling Methods:** We can approximate the integration in (19) by randomly drawing $(Y', \theta)$ from $p(Y, Y', \theta)$ and thereby performing inference about the distribution of $Y'$. Various methods to carry out such samplings exist. Typically one uses Markov Chain Monte Carlo (MCMC) methods. See [47] for details and further references.

The advantage of such methods is that given sufficient computational resources, we are able to obtain a very good estimate on the distribution of $Y'$. Furthermore the quality of the estimate keeps on increasing as we wait for more samples. The obvious downside is that we will not obtain a closed form analytic expression for a density. Furthermore sampling methods can be computationally quite expensive, especially if we wish to predict for a large amount of data.

**Variational Methods:** The reason why we had to resort to approximating (19) was that the integrand did not lend itself to a closed form solution of the integral. However, it may be that for a modified version of $p(Y, Y'|\theta)$ we might be able to perform the integral.

One option is to use a normal distribution $\mathcal{N}(\mu, \Sigma)$, where the mean $\mu$ coincides with the mode of $p(Y, Y'|\theta)p(\theta)$ with respect to $\theta$, and to use the second derivative of $-\ln p(Y, Y'|\theta)p(\theta)$ at $\theta = \mu$ for the variance $\sigma$. This is often referred to as the *Laplace Approximation.* We set (see for instance [40])

$$\theta|(Y, Y') \sim \mathcal{N}(E[\theta|(Y, Y')], \Sigma^{-1}) \text{ where } \Sigma = -\partial_\theta^2 \left[\ln p(\theta|(Y, Y'))\right]\Big|_\mu . \quad (20)$$

The advantage of such a procedure is that the integrals remain tractable. This is also one of the reasons why normal distributions enjoy a high degree of popularity in Bayesian methods. Besides, the normal distribution is the least informative distribution (largest entropy) among all distributions with bounded variance [7].

As Figure 3 indicates, a single Gaussian may not always be sufficient to capture the important properties of $p(Y, Y'|\theta)p(\theta)$. A more elaborate *parametric* model $q_\phi(\theta)$ of $p(\theta|Y, Y')$, such as a mixture of Gaussian densities, can then be used to improve the approximation of (19). A common strategy is to resort to variational methods. The details are rather technical and go beyond the scope of this section. The interested reader is referred to [36] for an overview, and to [4] for an application to the Relevance Vector Machine of Section 6. The following theorem describes the basic idea.

**Theorem 1 (Variational Approximation of Densities).** *Denote by $\theta, Y$ random variables with corresponding densities $p(\theta, Y), p(\theta|Y)$, and $p(\theta)$. Then for any density $q(\theta)$, the following bound holds;*

$$\ln p(Y) = \int_\theta \ln \frac{p(\theta, Y)}{q(\theta)} q(\theta) d\theta - \int_\theta \ln \frac{p(\theta|Y)}{q(\theta)} q(\theta) df \geq \int_\theta \ln \frac{p(\theta, Y)}{q(\theta)} q(\theta) d\theta. \quad (21)$$

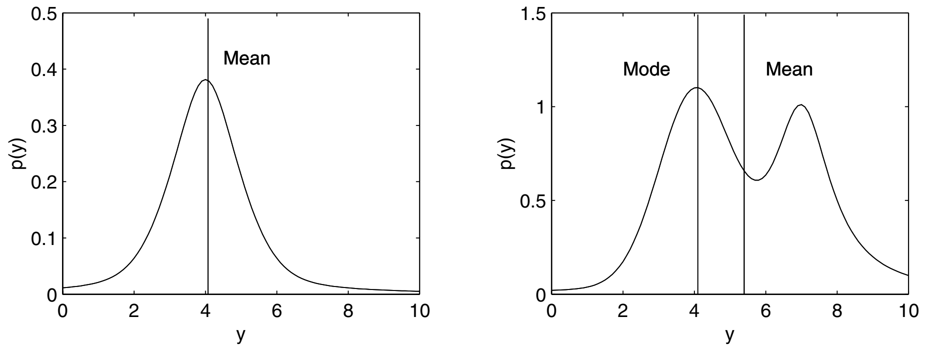

**Fig. 3.** Left: The mode and mean of the distribution coincide, hence the MAP approximation is satisfied. Right: For multi-modal distributions, the MAP approximation can be arbitrarily bad.

*Proof.* We begin with the first equality of (21). Since $p(\theta, Y) = p(\theta|Y)p(Y)$, we may decompose

$$\frac{p(\theta, Y)}{q(\theta)} = \ln p(Y) + \ln \frac{p(\theta|Y)}{q(\theta)}. \tag{22}$$

Additionally, $-\int_\theta \ln \frac{p(\theta|Y)}{q(\theta)} q(\theta) d\theta = \mathrm{KL}(p(\theta|Y)\|q(\theta))$ is the Kullback-Leibler divergence between $p(\theta|Y)$ and $q(\theta)$ [7]. The latter is a nonnegative quantity which proves the second part of (21).

The true posterior distribution is usually $p(\theta|Y)$, and $q(\theta)$ an approximation of it. The practical advantage of (21) is that

$$L := \int_\theta \ln \frac{p(\theta, Y)}{q(\theta)} q(\theta) d\theta$$

can often be computed more easily, at least for simple enough $q(\theta)$. Furthermore, by maximizing $L$ via a suitable choice of $q$, we maximize a lower bound on $\ln p(\theta)$.

**Expectation Maximization:** Another method for approximating $p(Y, Y')$ was suggested in [10], namely that one maximizes the integrand in (19) jointly over the unknown variable $Y'$ and the latent variable $\theta$. While this is clearly not equivalent to solving the integral, there are many cases where one may hope that the maximum of the *integrand* will not differ too much from the location of the mean of $p(y)$.

Furthermore we gain an interpretation of a possibly plausible value of $\theta$ in conjunction with $Y$. This could be the noise level of the estimation process, certain parameters of the function class such as the degree of smoothness, etc. Several options are at our disposal when it comes to maximizing $p(Y, Y'|\theta)p(\theta)$ with respect to $\theta$ and $Y'$.

Firstly, we could use a function maximization procedure directly on the joint distribution. Instead, [10] propose the so-called Expectation Maximization algorithm, which leads to a local maximum in a very intuitive fashion. Ignoring proof details we proceed as follows. Define

$$Q(\theta) := \mathbf{E}_{Y'}\left[\log p(Y'|Y,\theta)\right] + \log p(\theta) \tag{23}$$

where the expectation is taken over $p(Y'|Y,\hat{\theta})$ for a previously chosen value $\hat{\theta}$. Computing $Q$ is commonly referred as the *Expectation*-step. Next we find the maximum of $Q$ and replace $\hat{\theta}$ by it, that is

$$\hat{\theta} \leftarrow \operatorname*{argmax}_{\theta} Q(\theta). \tag{24}$$

This is commonly known as the *Maximization*-step. There exist many special cases where these calculations can be carried out in closed form, most notably distributions derived from the exponential family. [10] show that iteration of this procedure will converge to a local maximum of $p(Y,Y',\theta)$. However, it is not clear, how good the local maximum will be. After all, it depends on the initial guess of $\hat{\theta}$.
An alternative is to modify the definition of $Q(\theta)$ such that $\theta$ is used both for the conditional expectation and as an argument of the expectation itself (in other words, we merge $\hat{\theta}$ and $\theta$ into one variable). In this way we are still guaranteed to find a joint local maximum of $Y'$ and $\theta$ through maximizing a function of $\theta$ only, yet the reduced number of parameters can be beneficial for a general purposed function optimizer. We will come back to this observation in Section 3.

## 2.4 Likelihood, Priors, and Hyperpriors

Recall $p(Y,Y') = \int p(Y,Y'|\theta)p(\theta)$. This means that we weigh the values $p(Y,Y'|\theta)$, which tell us how well $(Y,Y')$ fits our model parameterized by $\theta$, by $p(\theta)$ according to our *prior knowledge* of the possible value of $\theta$.

For instance, if $p(Y,Y'|\theta)$ describes a jointly normal distribution of iid random variables with mean $\mu$ and variance $\sigma$, that is $\theta := (\mu,\sigma)$, $p(\theta)$ models our prior knowledge about the occurrence of a particular $(\mu,\sigma)$-pair. For instance, we might know that the variance never exceeds a certain value and that the mean is always positive.

Quite often, however, we may not even be sure about the specific form of $p(\theta)$ either, in which case we assume a distribution over possible priors, that is

$$p(\theta) = \int_{\omega} p(\theta|\omega)p(\omega). \tag{25}$$

Here $\omega$ is a so-called hyperprior describing the uncertainty in $\theta$. In summary, we have the following dependency model of $(Y,Y')$:

$$\xrightarrow{\quad p(\omega) \quad} \omega \xrightarrow[p(\theta|\omega)p(\omega)]{p(\theta|\omega)} \theta \xrightarrow[p(Y,Y'|\theta)p(\theta|\omega)p(\omega)]{p(Y,Y'|\theta)} (Y,Y') \tag{26}$$

Hence, in order to obtain $p(Y, Y')$ we need to solve the integral

$$\int_{\omega,\theta} p(Y, Y', \omega, \theta) d\theta d\omega = \int_{\omega,\theta} p(Y, Y'|\omega) p(\omega|\theta) p(\theta) d\theta d\omega \tag{27}$$

and again, as in the previous section, we may resort to various methods for approximating (27). By far the most popular method is to maximize the integrand and to obtain, what is commonly referred to as a MAP2 approximation:

$$\omega_{\mathrm{MAP}}(Y, \theta) := \underset{\omega}{\operatorname{argmax}}\, p(Y, \theta|\omega) p(\theta|\omega) p(\omega). \tag{28}$$

If possible, one integrates out the $\theta$ by solving the inner of the two integrals in (27) to obtain

$$\omega_{\mathrm{MAP}}(Y) := \underset{\omega}{\operatorname{argmax}}\, p(Y|\omega) p(\omega) = \underset{\omega}{\operatorname{argmax}} \int_\theta p(Y, Y'|\theta) p(\theta|\omega) d\theta. \tag{29}$$

In other words, $\theta$ assumes the role of the new prior, where $\omega$ is integrated out into $p(Y, Y'|\theta)$. Such an approach will be used in Section 6 to obtain numerically attractive optimization methods for estimation.

## 3 Gaussian Processes

Gaussian Processes are based on the "prior" assumption that adjacent observations should convey information about each other. In particular, it is assumed that the observed variables are normal, and that the coupling between them takes place by means of the covariance matrix of a normal distribution.

It turns out that this is a convenient way of extending Bayesian modeling of linear estimators to nonlinear situations (cf. [82, 80, 63]). Furthermore, it represents the counterpart of the "kernel trick" in methods minimizing the regularized risk. We present the basic ideas, and relegate details on efficient implementation of the optimization procedure required for inference to Section 4.

So far we only assumed that we have observations $Y$ and a density $p(Y)$, possibly $p(Y|\theta)$ that was given to us independently of any other data. However, if we wish to perform regression or classification, the observations $Y$ will typically depend on some patterns $X$. In other words, data comes in $(x_i, y_i)$ pairs and given some novel patterns $x'_i$ we will wish to estimate $y'_i$ at those locations. Hence, we will introduce a conditioning on $X$ and $X'$ in our reasoning. Note that this does not affect any of the derivations above.

### 3.1 Correlated Observations

If we are observing $y_i$ at locations $x_i$, it is only natural to assume that these values are correlated, depending on their location $x_i$. Indeed, if this were not the case, we would not be able to perform inference, since by definition, independent random variables $y_i$ do not depend on other observations $y_j$.

In fact, we make a rather strong assumption regarding the distribution of the $y_i$, namely that they form a normal distribution with mean $\mu$ and covariance matrix $K$.[4] We could of course assume *any* arbitrary distribution; most other settings, however, result in inference problems that are rather expensive to compute. Furthermore, as Theorem 5 will show, there exists a large class of assumptions on the distribution of $y_i$ that have a normal distribution as their limit.

We begin with two observations, $y_1$ and $y_2$, for which we assume zero mean $\mu = (0, 0)$ and covariance $K = \begin{bmatrix} 1 & 3/4 \\ 3/4 & 3/4 \end{bmatrix}$. Figure 4 shows the corresponding density of the random variables $y_1$ and $y_2$. Now assume that we observe $y_1$. This gives us further information about $y_2$, which allows us to state the conditional density[5]

$$p(y_2|y_1) = \frac{p(y_1, y_2)}{p(y_1)}. \tag{30}$$

Once the conditional density is known, the mean of $y_2$ need no longer be 0, and the variance of $y_2$ is decreased. In the example above, the latter becomes $\frac{3}{16}$ instead of $\frac{3}{4}$ — we have performed *inference* from the observation $y_1$ to obtain possible values of $y_2$.

In a similar fashion, we may infer the distribution of $y_i$ based on more than two variables, provided we know the corresponding mean $\mu$ and covariance matrix $K$. This means that $K$ determines how closely the prediction relates to the previous observations $y_i$. In the following section, we formalize the concepts presented here and show how such matrices $K$ can be generated efficiently.

## 3.2 Definitions and Basic Notions

Assume we are given a distribution over observations $y_i$ at locations $x_1, \ldots, x_m$. Rather than directly specifying that the observations $y_i$ are generated from an underlying functional dependency, we simply assume that they are generated by a Gaussian Process. Loosely speaking, Gaussian processes allow us to extend the notion of a set of random variables to random functions. More formally, we have the following definition:

**Definition 1 (Gaussian Process).** *Denote by $y(x)$ a stochastic process parameterized by $x \in \mathcal{X}$ ($\mathcal{X}$ is an arbitrary index set). Then $y(x)$ is a Gaussian process if for any $m \in \mathbb{N}$ and $\{x_1, \ldots, x_m\} \subset \mathcal{X}$, the random variables $(y(x_1), \ldots, y(x_m))$ are normally distributed.*

[4] Note that we now use $K$ to denote the covariance matrix. This is done for consistency with the literature on Reproducing Kernel Hilbert Spaces and Support Vector Machines, where $K$ denotes the kernel matrix. We will see that $K$ plays the same role in Gaussian Processes as the kernel matrix plays in the other settings.

[5] A convenient trick to obtain $p(y_2|y_1)$ for normal distributions is to consider $p(y_1, y_2)$ as a function only of $y_2$, while keeping $y_1$ fixed at its observed value. The linear and quadratic terms then completely determine the normal distribution in $y_2$.

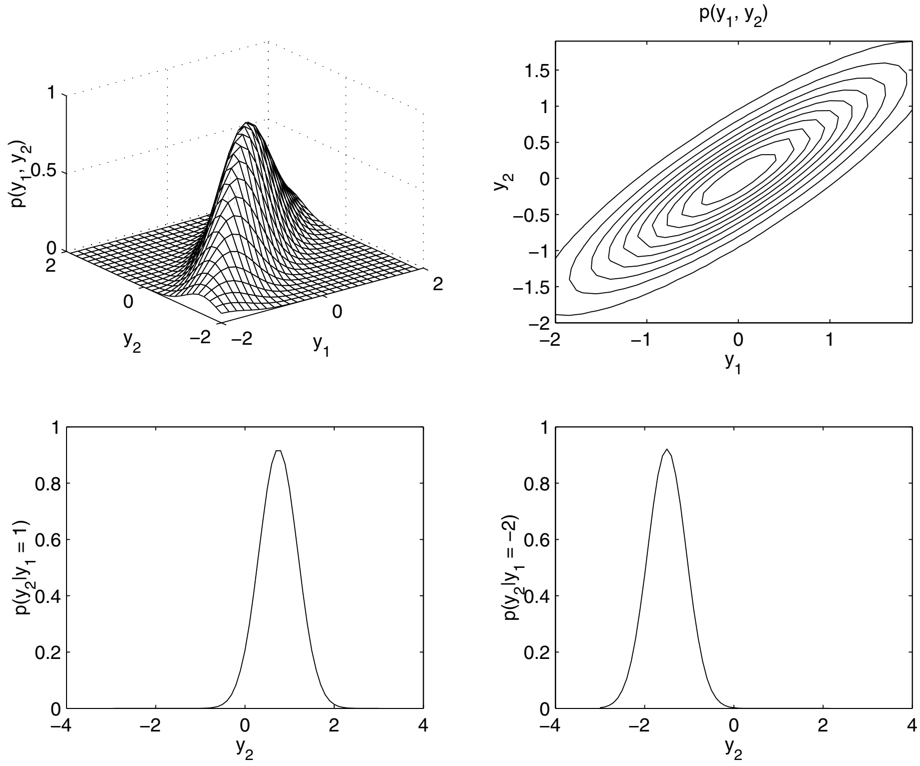

**Fig. 4.** Normal distribution with two variables. Top left: normal density $p(y_1, y_2)$ with zero mean and covariance $K$; Top right: contour plot of $p(y_1, y_2)$; Bottom left: Conditional density of $y_2$ when $y_1 = 1$; Bottom left: Conditional density of $y_2$ when $y_1 = -2$. Note that in the last two plots, $y_2$ is normally distributed, but with nonzero mean.

We denote by $k(x, x')$ the function generating the covariance matrix

$$K := \operatorname{cov}\{y(x_1), \dots, y(x_m)\}, \tag{31}$$

and by $\mu$ the mean of the distribution. We also write $K_{ij} = k(x_i, x_j)$. This leads to

$$(y(x_1), \dots, y(x_m)) \sim \mathcal{N}(\mu, K) \text{ where } \mu \in \mathbb{R}^m. \tag{32}$$

*Remark 1 (Gaussian Processes and Positive Definite Matrices).* The function $k(x, x')$ is well defined, symmetric, and the matrix $K$ is positive definite, that is, none of its eigenvalues is negative.

*Proof.* We first show that $k(x, x')$ is well defined. By definition,

$$[\operatorname{cov}\{y(x_1), \dots, y(x_m)\}]_{ij} = \operatorname{cov}\{y(x_i), y(x_j)\}. \tag{33}$$

Consequently, $K_{ij}$ is only a function of *two* arguments ($x_i$ and $x_j$), which shows that $k(x, x')$ is well defined.

It follows directly from the definition of the covariance that $k$ is symmetric. Finally, to show that $K$ is positive definite, we have to prove for any $\alpha \in \mathbb{R}^m$ that the inequality $\alpha^\top K \alpha \geq 0$ holds. This follows from

$$0 \leq \operatorname{Var}\left(\sum_{i=1}^{m} \alpha_i y(x_i)\right) = \alpha^\top \left[\operatorname{cov}\{y(x_i), y(x_j)\}\right] \alpha = \alpha^\top K \alpha. \tag{34}$$

Thus $K$ is positive definite and the function $k$ is an admissible kernel.

Note that even if $k$ happens to be a smooth function (this turns out to be a reasonable assumption), the actual realizations $y(x)$, as drawn from the Gaussian process, need not be smooth at all. In fact, they may be even pointwise discontinuous.

Let us have a closer look at the prior distribution resulting from these assumptions. The standard setting is $\mu = 0$, which implies that we have no prior knowledge about the particular value of the estimate, but assume that small values are preferred. Then, for a given set of $(y(x_1), \ldots, y(x_m)) =: Y$, the prior density function $p(Y|X)$ is given by

$$p(Y|X) = (2\pi)^{-\frac{m}{2}} (\det K)^{-\frac{1}{2}} \exp\left(-\frac{1}{2} Y^\top K^{-1} Y\right). \tag{35}$$

In most cases, we try to avoid inverting $K$. By a simple substitution,

$$Y := K\boldsymbol{\alpha}, \tag{36}$$

we have $\boldsymbol{\alpha} \sim \mathcal{N}(0, K^{-1})$, and consequently

$$p(\boldsymbol{\alpha}|X) = (2\pi)^{-\frac{m}{2}} (\det K)^{-\frac{1}{2}} \exp\left(-\frac{1}{2} \boldsymbol{\alpha}^\top K \boldsymbol{\alpha}\right). \tag{37}$$

Taking logs, we see that this term is identical to penalty term arising from the regularized risk framework (cf. the chapter on Support Vectors and [62, 77, 75, 26]). This result thus connects Gaussian process priors and estimators using the Reproducing Kernel Hilbert Space framework: Kernels favoring smooth functions translate immediately into covariance kernels with similar properties in a Bayesian context.

## 3.3 Simple Hypotheses

Let us analyze in more detail which functions are considered simple by a Gaussian process prior. As we know, hypotheses of low complexity correspond to vectors $Y$ for which $Y^\top K^{-1} Y$ is small. This is in particular the case for the (normalized) eigenvectors $v_i$ of $K$ with large eigenvalues $\lambda_i$, since

$$K v_i = \lambda_i v_i \text{ yields } v_i^\top K^{-1} v_i = \lambda_i^{-1}. \tag{38}$$

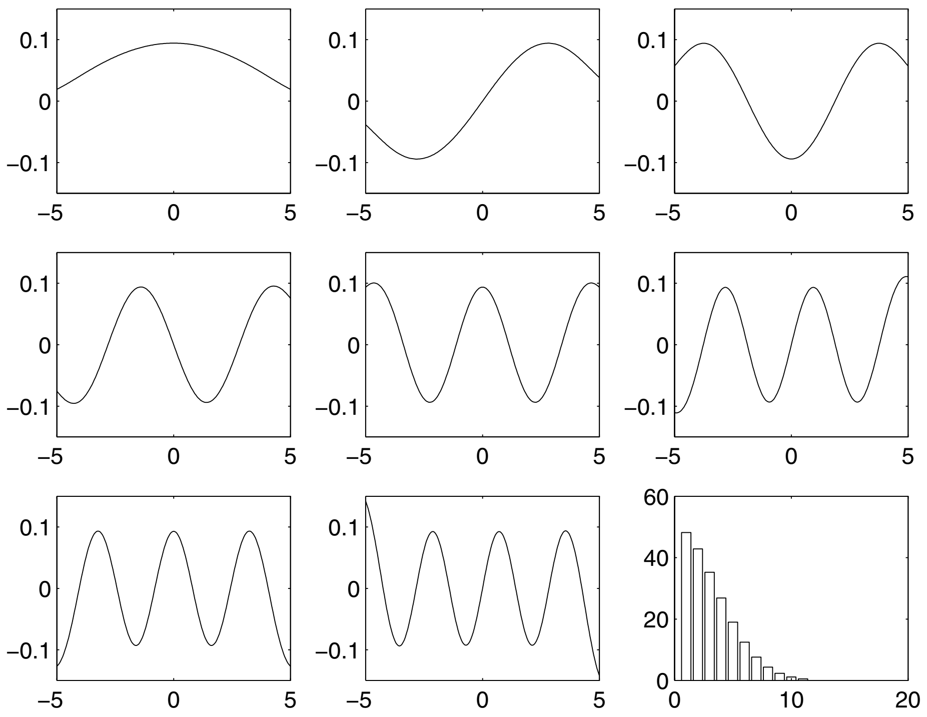

**Fig. 5.** Hypotheses corresponding to the first eigenvectors of a Gaussian kernel of width 1 over a uniform distribution on the interval $[-5, 5]$. From top to bottom and from left to right: The functions corresponding to the first eight eigenvectors of $K$. Lower right: the first 20 eigenvalues of $K$. Note that most of the information about $K$ is contained in the first 10 eigenvalues. The plots were obtained by computing $K$ for an equidistant grid of 200 points on $[-5, 5]$. We then computed the eigenvectors $\mathbf{e}$ of $K$, and plotted them as the corresponding function values (this is possible since for $\boldsymbol{\alpha} = \mathbf{e}$ we have $K\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}$).

In other words, the estimator is biased towards solutions with small $\lambda_i^{-1}$. This means that the spectrum and eigensystem of $K$ represent a practical means of actually *viewing* the effect a certain prior has on the degree of smoothness of the estimates. Let us consider a practical example: for a Gaussian covariance kernel

$$k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\omega^2}\right), \tag{39}$$

where $\omega = 1$, and under the assumption of a uniform distribution on $[-5, 5]$, we obtain the functions depicted in Figure 5 as simple base hypotheses for our estimator. Note the similarity to a Fourier decomposition: this means that the kernel has a strong preference for slowly oscillating functions.

The Gaussian Process setting also allows for a simple connection to parametric models of uncertainty (see also [80]). For instance, assume that the observations $y$ are derived from the patterns $x$ via the functional dependency

$$y(x) = \sum_{i=1}^{n} \beta_i f_i(x), \text{ where } \beta \sim \mathcal{N}(0, \Sigma). \tag{40}$$

Clearly the random variables $y$ are jointly normal, hence stem from an underlying Gaussian Process. We can calculate the covariance function $k$ as follows. Let $\mathbf{f}(x) := (f_1(x), \ldots, f_n(x))$, then

$$k(x, x') = \operatorname{Cov}(y(x), y(x')) = \mathbf{f}(x)^\top \Sigma \mathbf{f}(x'). \tag{41}$$

In other words, starting from a parametric model, where we would want to estimate the coefficients $\beta$ we arrived at a Gaussian Process with covariance function $\mathbf{f}(x)^\top \Sigma \mathbf{f}(x')$.

One special case is of interest: set $f_i(x) = (x)_i$, that is, $f_i(x)$ encodes the $i$-th coordinate of $x$, and $\Sigma = \mathbf{1}$. Here $k(x, x') = \langle x, x' \rangle$. This is the simplest Gaussian Process kernel possible. Other kernels include

$$k(x, x') = \exp(-\omega \|x - x'\|) \text{ (Laplacian kernel)} \tag{42}$$
$$k(x, x') = (\langle x, x' \rangle + c)^p \text{ with } c > 0 \text{ (Polynomial kernel)} \tag{43}$$

and the Gaussian RBF kernel of (39). For further details on the choice of kernels see [62, 25, 78, 31, 52, 77] and the references therein.

## 3.4 Regression

Let us put the previous discussion to practical use. For the sake of simplicity, we begin with regression (we study the classification setting in the next section).

The natural assumption is that the observations $Y$ are generated by the Gaussian Process with covariance matrix $K$ and mean $\mu$. Following the reasoning of Example 1 we can infer novel $y_i'$, given $x_i$ via $Y' \sim \mathcal{N}(\Sigma', \mu')$ where

$$K_{\text{cond}} = K_{Y'Y'} - K_{Y'Y} K_{YY}^{-1} \Sigma_{YY'} \tag{44}$$
$$\mu_{\text{cond}} = \mu_{Y'} + K_{Y'Y} K_{YY}^{-1} (Y - \mu_Y) \tag{45}$$

This means that the variance is reduced from $K_{Y'Y'}$ to a degree controlled by both the correlation between $Y$ and $Y'$ (via $K_{Y'Y}$), and the inherent degree of variability in $Y$ (via $K_{YY}^{-1}$). The more certain we can be about $Y$, the more this certainty carries over to $Y'$. Likewise, the default estimate for the mean of $Y'$, namely $\mu_{Y'}$ is corrected by $K_{Y'Y} K_{YY}^{-1} (Y - \mu_Y)$, that is, the deviation of $Y$ from its default estimate $\mu_Y$, weighted by the correlation between the random variables $Y$ and $Y'$.

Furthermore note that for the purpose of inferring the mean we only need to store $\boldsymbol{\alpha} := K_{YY}^{-1}(Y - \mu_Y)$ and hence $(Y - \mu_Y) = K_{YY}\boldsymbol{\alpha}$. This is similar to (36), where we used $\boldsymbol{\alpha}$ to establish a connection between the regularized risk functional and prior probabilities.

Let us get back to the case where $k(x, x') = \langle x, x' \rangle$, that is, the linear covariance kernel. Here we can rewrite (44) as

$$K_{\text{cond}} = X'X'^\top - (X'X^\top)(XX^\top)^{-1}(XX'^\top) = X'(\mathbf{1} - P_X)X'^\top \tag{46}$$

where $P_X$ is the projection on the space spanned by $X$. This means that the larger $X$ grows, the more reliably we will be able to infer $Y'$ for a given $X'$. In the case where $X$ spans the entire space (if $x \in \mathbb{R}^n$ it may suffice if $|X| = n$), $P_X$ is the identity and consequently $K_{\text{cond}} = 0$. In other words, we can perform inference with *certainty.*

The avid reader will notice that prediction with certainty can pose a problem if our model is not quite exact or if the data itself is fraught with measurement errors or noise. For instance, if we were to observe some $Y$ which do *not* lie in the $n$-dimensional subspace spanned by $X$, we would have to conclude that such $Y$ must never occur. In other words, we have a modeling problem. One way to address this issue is to replace the (apparently) unsuitable kernel $k(x,x') = \langle x, x' \rangle$ by one which guarantees that $K$ has always full rank. This, however, is rather cumbersome, since the question whether $K$ is nondegenerate depends on both the data and on the covariance function $k(x,x')$.

A more elegant way to ensure that our statistical model is better suited to the task is to introduce an extended dependency model using latent variables as follows:

$$\begin{tikzcd} {} \arrow[r, "p(X)"] & X \arrow[r, "p(\boldsymbol{\theta}|X)", "p(\boldsymbol{\theta}|X)p(X)"'] & \boldsymbol{\theta} \arrow[r, "p(Y|\boldsymbol{\theta})", "p(Y|\boldsymbol{\theta})p(\boldsymbol{\theta}|X)p(X)"'] & Y \end{tikzcd}$$

This means that the random variables $X$ and $Y$ are conditionally independent, given $\boldsymbol{\theta}$. If we wish to infer $Y'$ from $X, X', Y'$ this means that we need to integrate out the *latent* variable $\boldsymbol{\theta}$, in the same fashion as we did in (26) and as discussed in Section 2.3.

In regression, a popular setting for $p(Y|\boldsymbol{\theta})$ is to assume that we have additive noise, i.e., $Y = \boldsymbol{\theta} + \boldsymbol{\xi}$, where the $\xi_i$ are drawn from some distribution (Gaussian, Laplacian, Huber's robust, etc.), such as the ones given in Figure 1. And again, as discussed in Section 2.3, some of the integrals arising from the elimination of $\boldsymbol{\theta}$ may be easily solvable, or we may need to resort to further approximations.

**Additive Normal Noise:** We begin with the simple case, where $\xi_i \sim \mathcal{N}(0, \sigma^2)$. In this situation $Y \sim \mathcal{N}(\mu, \sigma^2\mathbf{1} + K)$, since $Y$ is the sum of two independent normal random variables. Consequently we can re-use (44) and (45), the only difference being that now instead of $k(x,x')$ we use $k(x,x') + \sigma^2\delta_{x,x'}$ as the covariance function. Figure 6 gives an example of regression with additive normal noise.

Specializing (44) and (45) to the estimation of $y(x)$ at only *one* new location and assuming $\mu = 0$ we obtain (using $\mathbf{k}(x) := (k(x_1,x), \dots, k(x_m,x))$)

$$K_{\text{cond}} = \sigma^2 + k(x,x) - \mathbf{k}(x)^\top (K_{YY} + \sigma^2\mathbf{1})^{-1}\mathbf{k}(x) \tag{47}$$

$$\mu_{\text{cond}} = \mathbf{k}(x)^\top (K_{YY} + \sigma^2\mathbf{1})^{-1} Y = \mathbf{k}(x)^\top \boldsymbol{\alpha} \text{ where } \boldsymbol{\alpha}(K_{YY} + \sigma^2\mathbf{1})^{-1} Y. \tag{48}$$

Note that the mean is a linear combination of kernel functions $k(x_i, x)$. Moreover, if we were to estimate $y(x)$ for an element of $x$, say $y(x_i)$, we would obtain

$$\mathbf{k}(x_i)^\top (K_{YY} + \sigma^2\mathbf{1})^{-1} Y = \left[K_{YY}(K_{YY} + \sigma^2\mathbf{1})^{-1} Y\right]_i. \tag{49}$$

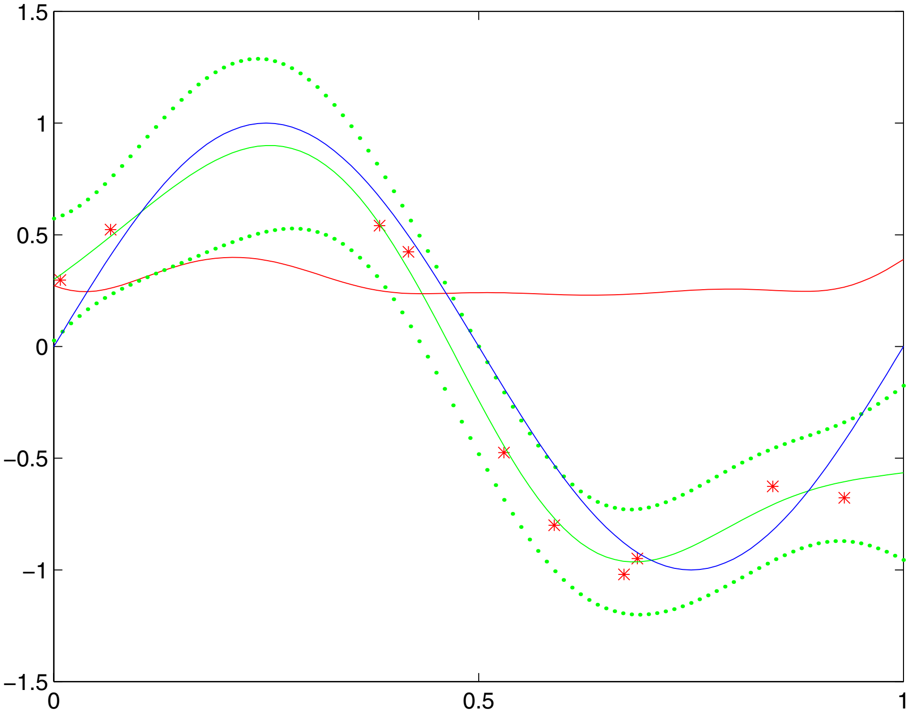

**Fig. 6.** Gaussian Process regression with additive normal noise. The stars denote the observations, dotted lines correspond to the $\sigma$-confidence intervals with the prediction in between, and the solid line crossing the boundaries is the sine function, which, with additive normal noise, was used to generate the observations. For convenience we also plot the width of the confidence interval.

If $\sigma^2$ were 0 we would obtain $y_i$, however, with $\sigma^2 > 0$ we end up *shrinking* $y_i$ towards 0, similarly to the shrinkage estimator of James and Stein [33].

**Other Additive Noise:** While we may in general not be able to integrate out $\boldsymbol{\theta}$ from $p(Y|\boldsymbol{\theta})$, the noise models of regression typically allow one to perform several simplifications when it comes to estimation. For convenience we will in the following split up the latent variable into $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ corresponding to $Y$ and $Y'$. Moreover, we will assume that the additive noise has zero mean.[6]

Recall from Section 2.3 that an approximation to marginalization is to maximize the joint density $p(Y', \boldsymbol{\theta}, \boldsymbol{\theta}', Y, X, X')$ with respect to $(Y', \boldsymbol{\theta}, \boldsymbol{\theta}')$. Using the fact that the random variables $y_i$ are drawn iid and that $Y, Y'$ are conditionally independent of $X, X'$ given $\boldsymbol{\theta}, \boldsymbol{\theta}'$ we arrive at

[6] In the nonzero mean case we simply add the offset to $\mu$, which effectively reduces the additive noise to zero mean.

$$p(Y', \boldsymbol{\theta}, \boldsymbol{\theta}', Y, X, X') = p(Y'|\boldsymbol{\theta}')p(Y|\boldsymbol{\theta})p(\boldsymbol{\theta}'|\boldsymbol{\theta}, X, X')p(\boldsymbol{\theta}|X)p(X, X'). \quad (50)$$

Note that $Y'$ appears only in $p(Y'|\boldsymbol{\theta})$. If $p(y_i|\theta_i)$ has its mode for $y_i = \theta_i$, we know that $Y' = \boldsymbol{\theta}$ maximizes (50) and we only need to care about the remainder

$$p(Y' = \boldsymbol{\theta}'|\boldsymbol{\theta}')p(Y|\boldsymbol{\theta})p(\boldsymbol{\theta}'|\boldsymbol{\theta}, X, X')p(\boldsymbol{\theta}|X) \quad (51)$$

and maximize it with respect to $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$. A further simplification occurs if $p(Y' = \boldsymbol{\theta}'|\boldsymbol{\theta}')$ is constant, which is typically the case (e.g., if we have the same additive noise for all observations). Then the maximization with respect to $\boldsymbol{\theta}'$ can be carried out by maximizing $p(\boldsymbol{\theta}'|\boldsymbol{\theta}, X, X')$. Note that the latter is Gaussian in $\boldsymbol{\theta}'$, so the maximum is obtained for the mean $\mu_{\text{cond}}$ of the corresponding normal distribution, as given by (44) and (45). Moreover, $p(\boldsymbol{\theta}' = \mu_{\text{cond}}|\boldsymbol{\theta}, X, X')$ is constant, if we consider $\mu_{\text{cond}}$ as a function of $\boldsymbol{\theta}$. This means that we now reduced the problem to the one of maximizing

$$p(Y|\boldsymbol{\theta})p(\boldsymbol{\theta}|X). \quad (52)$$

The latter depends on the training data $X, Y$ alone, which makes the whole approach computationally very attractive. In summary, the following steps were needed in reducing (50) to (52):

1. The conditional distribution $p(y_i|\theta_i)$ is peaked for $y_i = \theta_i$.
2. $p(y_i = \theta_i|\theta_i)$ is constant, when considered as a function of $\theta_i$.
3. We predict $\boldsymbol{\theta}' = \mu_{\text{cond}}$ for known $\boldsymbol{\theta}$.
4. The maximizer in $\boldsymbol{\theta}$ is found by maximizing the posterior probability $p(\boldsymbol{\theta}|X) \propto p(Y|\boldsymbol{\theta})p(\boldsymbol{\theta}|X)$.

We shall see in the next section that even if some of the above assumptions do not lead to optimality, we may still be able to obtain reasonably good estimates.

In situations where neither of the two special cases discussed above applies we will need to resort to a reasoning which is very similar to the one used in Gaussian Process classification. Since such situations are rather rare we will not discuss them in further detail.

## 3.5 Classification

The main difference to regression is that in classification the observations $y_i$ are part of a discrete set $\mathcal{Y}$, which we will either denote by $\mathcal{Y} = \{\pm 1\}$ in the case of binary classification and $\mathcal{Y} = \{1, \ldots, N\}$ for multiclass problems. It is clear that in such a situation we will *need* a conditional probability $p(y_i|\theta_i)$ if we are to transform the problem into one involving Gaussian Processes. This will lead to various models, using the logistic transfer function (14), the Probit (16), and their multiclass extensions.

**Binary Case:** Here $\theta_i$ plays the role of a parameter responsible for the calibration of the conditional probabilities $\mathrm{P}(y_i = 1|X)$ and $\mathrm{P}(y_i = -1|X)$ and we have

$$p(Y|X) = \int \left[\prod_{i=1}^{m} p(y_i|\theta_i)\right] p(\boldsymbol{\theta}|X)d\boldsymbol{\theta}. \tag{53}$$

$Y$ being a discrete random variable, this gives us the *probability* of observing $Y$, given $X$. The latter makes classification slightly easier than regression: provided we are able to solve (53), we immediately know the confidence of the random variables $Y$ arising from it. Thus, $\mathrm{P}(y_i = 1|X)$ not only tells us whether the estimator classifies $x_i$ as $+1$ or $-1$, but also the probability of obtaining these labels. Therefore, calculations regarding the variation of $Y$ are not quite as important as they were for regression.

**Multiclass Classification:** While one could in theory design some $p(y_i|\theta_i)$ which allows for multiple discrete values of $y_i$ and assigns corresponding probabilities, it is far from clear how such a goal can be best achieved. Instead, one typically uses vector valued $\theta_i \in \mathbb{R}^N$ such that each coordinate $[\theta_i]_j$ is related to the probability of class $j$ occurring. To derive our model we begin with the observation that for logistic regression

$$p(y_i = 1|\theta_i) = \frac{\exp(\frac{1}{2}\theta_i)}{\exp(\frac{1}{2}\theta_i) + \exp(-\frac{1}{2}\theta_i)} \tag{54}$$

$$p(y_i = -1|\theta_i) = \frac{\exp(-\frac{1}{2}\theta_i)}{\exp(\frac{1}{2}\theta_i) + \exp(-\frac{1}{2}\theta_i)}. \tag{55}$$

In other words, the probability of class 1 or $-1$ occurring is proportional to $\exp(\frac{1}{2}[\theta_i]_1)$ and $\exp(\frac{1}{2}[\theta_i]_{-1})$ subject to a normalization constraint, where we defined the coordinate "$[\theta_i]''_{-1} := -\theta_i$.

From there the extension to more than two classes is straightforward: assume that the probability of class $j$ is proportional to $\exp(\frac{1}{2}[\theta_i]_j)$ and normalize such that all $p(y_i = j|\theta_i)$ sum up to 1, namely

$$p(y_i = j|\theta_i) = \frac{\exp\left(\frac{1}{2}[\theta_i]_j\right)}{\sum_{l=1}^{N} \exp\left(\frac{1}{2}[\theta_i]_l\right)}. \tag{56}$$

The default assumption is then that all $\theta_i$ are drawn from a Gaussian Process. Without any further knowledge about the relation between the various classes $j$, one typically assumes that all coordinates are drawn independently. An extension to uncertain labels $y_i$ is straightforward. Assume that we do not know the specific class $j$ but merely some probability $p_{ij}$ assessing whether pattern $x_i$ belongs to class $j$ or not. In this case we need to integrate over all $j$ to obtain

$$p(Y_{\text{uncertain}}|\boldsymbol{\theta}) = \prod_{i=1}^{m} \left(\sum_{j=1}^{N} p_{ij} p(j|\theta_i)\right) \tag{57}$$

as a replacement for $p(y_i = j|\theta_i)$ in the conditional probability $p(Y|\boldsymbol{\theta})$. This is closely related to uncertain multiclass settings from maximum margin theory. See [55] for further details.

To solve the inference problem arising from (19) or (50) in the classification case we will need to resort to approximations. Again, as before, we cannot solve the integral explicitly, so our main goal is to (approximately) maximize the joint density. For convenience, we only study the binary case.

Recall that $Y'$ appears in the joint density only via $p(Y'|\boldsymbol{\theta})$. Knowing that this is maximized with respect to $Y'$ if we set $y'_i = \operatorname{sgn}(\theta'_i)$ we could turn the overall problem into one of jointly maximizing over the continuous random variables $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$. Unfortunately the resulting optimization problem is rather ill behaved. Instead we resort to the EM algorithm or a related method, as described in Section 2.3. Here $Y'$ are the unknown labels and $\boldsymbol{\theta}, \boldsymbol{\theta}'$ the latent variables to be obtained jointly with $Y'$.

Taking the expectation over $Y'$ as given by $p(Y'|\boldsymbol{\theta}')$ we can write $Q(\theta)$ (see (23)) as follows

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}') = \mathbf{E}_{Y'}\left[\log p(Y'|\boldsymbol{\theta}')\right] + \log p(Y|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}, \boldsymbol{\theta}'|X, X') \tag{58}$$

$$= \sum_{i=1, y'_i \in \mathcal{Y}}^{m'} p(y'_i|\theta_i) \log p(y'_i|\bar{\theta}'_i) + \log p(Y|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}, \boldsymbol{\theta}'|X, X') \tag{59}$$

where $\bar{\theta}'_i$ denotes the value of $\theta'_i$ obtained from a previous estimate. One then proceeds as follows: at every step one maximizes $Q(\boldsymbol{\theta}, \boldsymbol{\theta}')$ and subsequently updates $\bar{\theta}'_i \leftarrow \theta'_i$ until convergence. [10] show that the algorithm converges to a local maximum of the joint density $p(Y', \boldsymbol{\theta}, \boldsymbol{\theta}'|Y, X, X')$.

Unfortunately, in practice, the local maximum achieved by this process is often not very good. However, a small modification of (58) leads to an expression which (empirically) tends to lead to better estimates, yet will also lead to a local maximum of the joint density. The idea is to remove $\bar{\theta}'_i$ from $Q$ and replace it by $\theta'_i$ directly. In this case we have

$$\tilde{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}') = \sum_{i=1, y'_i \in \mathcal{Y}}^{m'} p(y'_i|\theta_i) \log p(y'_i|\theta'_i) + \log p(Y|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}, \boldsymbol{\theta}'|X, X') \tag{60}$$

$$= -\sum_{i=1}^{m'} H(p(y_i|\theta'_i)) + \log p(Y|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}, \boldsymbol{\theta}'|X, X'), \tag{61}$$

where $H(p) = \sum_j p_j \log p_j$ is the entropy of $p$. This function can then be maximized jointly over $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ by standard nonlinear optimization methods. Note that the maxima of $\tilde{Q}$ and the fixed point of $Q$ have to coincide. Since we know that the latter are the (local) maxima of the joint density we know that $\tilde{Q}$, too, will yield maxima of the joint density.

From a Minimum Description Length point of view, minimizing $-\tilde{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}')$ can be viewed as minimizing the number of bits (we use base $e$ for convenience) needed to encode $(Y', \boldsymbol{\theta}, \boldsymbol{\theta}')$, given $Y, X, X'$: Firstly encoding $Y'$, given $\boldsymbol{\theta}'$ requires $H(p(Y|\boldsymbol{\theta}')$ bits. Secondly, for a given prior probability $p(\boldsymbol{\theta}, \boldsymbol{\theta}'|X, X')$, the two remaining terms in (60) describe the number of bits needed to encode these random variables with respect to a code using $\log p(Y|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}, \boldsymbol{\theta}'|X, X')$ bits. The latter is the Shannon optimal code length for random variables distributed according to the assumed prior distribution.

Note that quite commonly one ignores the terms dependent on $X', \boldsymbol{\theta}'$ when it comes to finding an estimate for $\boldsymbol{\theta}$, hence one effectively resorts to an optimization setting as described in Section 3.4, however without the justification that could be given in the additive-noise regression case. This tends to yield acceptable results, however, it is important to bear in mind that better estimates can be obtained if a joint maximization over all variables is carried out. Figure 7 gives an example on classification with and without taking $Y'$ into account.

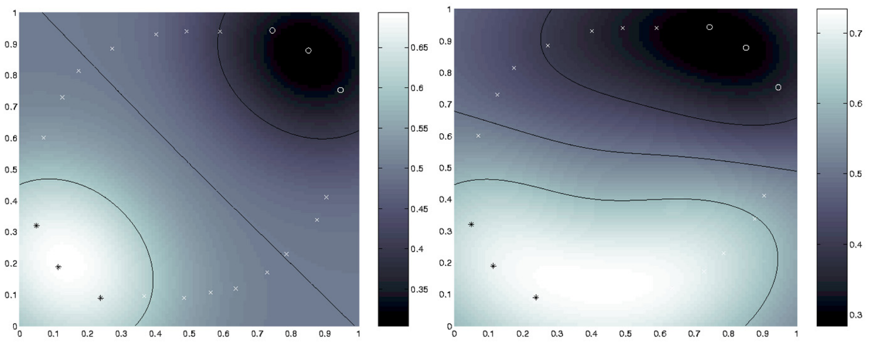

**Fig. 7.** Gaussian Process classification without (left) and with (right) knowledge of the test data. Circles and stars correspond to the respective classes, crosses are unlabeled observations. Note the errors introduced by ignoring the test data on the left figure.

## 3.6 Adjusting Hyperparameters for Gaussian Processes

More often than not, we will not know the exact amount of additive noise, the specific form of the covariance kernel, or other parameters beforehand. To address this problem, the hyperparameter formalism of Section 2.4 is needed.

However, unlike in the previous section, even an EM approach may be too costly, since expectations over the set of hyperparameters $\omega$, or over the set of further latent variables $\boldsymbol{\theta}, \boldsymbol{\theta}'$ are too costly to be carried out. Consequently, one of the few practical method available is that of coordinate descent, that is: a) optimize over $(\boldsymbol{\theta}, \boldsymbol{\theta}', Y')$ via the EM algorithm for a fixed $\omega$, b) maximize with respect to $\omega$ for a fixed $(\boldsymbol{\theta}, \boldsymbol{\theta}', Y')$, and repeat a) and b) until convergence occurs.

To avoid technicalities, we only discuss the special and somewhat simpler case of regression with additive Gaussian noise, since here the latent variables

$\boldsymbol{\theta}, \boldsymbol{\theta}'$ can be integrated out. We refer the reader to [84, 15, 11, 54] and the references therein for integration methods based on Markov Chain Monte Carlo approximations (see also [63] for a more recent overview).

More specifically, assume that both $K$ and $\sigma^2$ (the additive normal noise) are functions of $\omega$, so that

$$p(Y|\omega, X) = \frac{1}{\sqrt{(2\pi)^m \det(K + \sigma^2 \mathbf{1})}} \exp\left(-\frac{1}{2} Y^\top (K + \sigma^2)^{-1} Y\right) \qquad (62)$$

and $p(Y|X) = \int p(Y|\omega, X) p(\omega) d\omega$. In other words, (62) tells us how likely it is that we observe $\mathbf{y}$, if we know $\omega$. To maximize the integrand $p(Y|\omega, X)p(\omega)$ with respect to $\omega$ we require information about the gradient of (62) with respect to $\omega$. An explicit expression is given below

Since the logarithm is monotonic, we can equivalently minimize the negative log posterior, $\ln p(Y|\omega)p(\omega)$. With the shorthand $Q := K + \sigma^2 \mathbf{1}$, we obtain

$$\begin{aligned} &\partial_\omega \left[-\ln p(Y|\omega)p(\omega)\right] \\ &= \frac{1}{2}\partial_\omega (\ln \det Q) - \frac{1}{2}\partial_\omega \left[Y^\top Q^{-1} Y\right] - \partial_\omega \ln p(\omega) & (63) \\ &= -\frac{1}{2} \operatorname{tr}\left(Q^{-1} \partial_\omega Q\right) + \frac{1}{2} Y^\top Q^{-1} \left(\partial_\omega Q\right) Q^{-1} Y - \partial_\omega \ln p(\omega). & (64) \end{aligned}$$

Here (64) follows from (63) via standard matrix algebra [39]. Likewise, we could compute the Hessian of $\ln p(Y|\omega)p(\omega)$ with respect to $\omega$ and use a second order optimization method.[7]

If we assume a flat[8] hyperprior ($p(\omega) = \text{const.}$), optimization over $\omega$ simply becomes gradient descent in $-\ln p(Y|\omega)$; in other words, the term depending on $p(\omega)$ vanishes. Computing (64) is still very expensive numerically since it involves the inversion of $Q$, which is an $m \times m$ matrix.

One option to parameterize $K + \sigma^2 \mathbf{1}$ is to assume that the covariance kernel $k$ itself has been drawn from a Gaussian Process (in this case we need to restrict $k$ to to the cone of Mercer kernels). Such a setting can be optimized by a so-called superkernel expansion. See [70] for further details.

Finally, there exist numerous techniques, such as sparse greedy approximation methods, to alleviate this problem. We present a selection of these techniques in the following section. Additional detail on the topic of hyperparameter optimization can be found in Section 6, where hyperparameters play a crucial role in determining the sparsity of an estimate.

## 4 Implementation of Gaussian Processes

In this section, we discuss various methods to perform inference in the case of Gaussian process classification or regression. We begin with a general purpose

[7] This is rather technical, and the reader is encouraged to consult the literature for further detail [41, 54, 46, 18].

[8] Note that this is clearly an *improper* hyperprior, which may lead to overfitting.

technique, the *Laplace approximation*, which is essentially an application of Newton's method to the problem of minimizing the negative log-posterior density. Since it is a second order method, it is applicable as long as the log-densities have second order derivatives. Readers interested only in the basic ideas of Gaussian process estimation may skip the present section.

For classification with the logistic transfer function we present a variational method (Section 4.2), due to Jaakkola and Jordan [32], and Gibbs and MacKay [18, 16], a linear system of equations for optimization purposes.

Finally, the special case of regression in the presence of normal noise admits very efficient optimization algorithms based on the approximate minimization of quadratic forms (Section 4.3). We subsequently discuss the scaling behavior and approximation bounds for these algorithms. For convenience, we only study the problem of maximizing

$$p(\boldsymbol{\theta}|Y,X) \propto p(Y|\boldsymbol{\theta})p(\boldsymbol{\theta}|X) \tag{65}$$

ignoring the considerations about the test data $X'$, which we put forward in Section 3.5. An extension to these methods is in some cases straightforward (for the Expectation Maximization setting), and in other cases essentially impossible (for the direct MDL approach). So we skip both of them. Maximizing $p(\boldsymbol{\theta}|Y,X)$ is equivalent to minimizing $-\log p(Y|\boldsymbol{\theta}) - \log p(\boldsymbol{\theta}|X)$, since

$$\mathfrak{L}(\boldsymbol{\theta}) := -\log(\boldsymbol{\theta}|Y,X) = -\log p(\boldsymbol{\theta}|Y,X) = -\log p(Y|\boldsymbol{\theta}) - \log p(\boldsymbol{\theta}|X) + c, \tag{66}$$

for some constant $c \in \mathbb{R}$. $\mathfrak{L}(\boldsymbol{\theta})$ is commonly referred to as the negative log posterior and the remainder of this section is devoted to efficient methods of minimizing it.

## 4.1 Laplace Approximation

Note that for Gaussian Process regression with additive normal noise (66) becomes

$$\mathfrak{L}(\boldsymbol{\theta}) = \sum_{i=1}^{m} \frac{1}{2\sigma^2}(y_i - \theta_i)^2 + \frac{1}{2}\boldsymbol{\theta}K^{-1}\boldsymbol{\theta}^\top. \tag{67}$$

Minimization of (67) can be achieved by solving a quadratic optimization problem with its minimum $\hat{\boldsymbol{\theta}}$ at

$$\hat{\boldsymbol{\theta}} = (\sigma^2\mathbf{1} + K)^{-1}KY = K(\sigma^2\mathbf{1} + K)^{-1}Y \tag{68}$$

which is identical to the estimate obtained in (48). This means that for normal distributions seeking the mode of the density and performing a quadratic approximation at the mode is exact.

In general, however, the negative log posterior (66), is not quadratic, hence the minimum cannot be found analytically and typically we will not be able to study the variation of the estimate explicitly either. A possible solution is to make successive quadratic approximations of the negative log posterior, and minimize the latter iteratively. This strategy is referred to as the Laplace approximation[9] [71, 84, 63]; the Newton-Raphson method, in numerical analysis (see [72, 53]); or the Fisher scoring method, in statistics.

A necessary condition for the minimum of a differentiable function $g$ is that its first derivative be 0. For convex functions, this requirement is also sufficient. We approximate $g'$ linearly by

$$g'(x + \Delta x) \sim g'(x) + \Delta x g''(x), \text{ and hence } \Delta x = -\frac{g'(x)}{g''(x)}. \tag{69}$$

Substituting $\ln p(f|X, Y)$ into (69) and using the definitions

$$c := \left(-\partial_{\theta_1} \ln p(y_1|\theta_1), \ldots, -\partial_{\theta_m} \ln p(y_m|\theta_m)\right), \tag{70}$$

$$C := \operatorname{diag}\left(-\partial^2_{\theta_1} \ln p(y_1|\theta_1), \ldots, -\partial^2_{\theta_m} \ln p(y_m|\theta_m)\right), \tag{71}$$

we obtain the following update rule for $\boldsymbol{\alpha}$ (where $\boldsymbol{\theta} = K\alpha$),

$$\boldsymbol{\alpha}_{\text{new}} = (KC + \mathbf{1})^{-1}(KC\boldsymbol{\alpha}_{\text{old}} - c). \tag{72}$$

While (72) is usually an efficient way of finding a maximizer of the log posterior, it is far from clear that this update rule is always convergent (to prove the latter, we would need to show that the initial guess of $\boldsymbol{\alpha}$ lies within the radius of attraction [53, 13, 19, 38]. Nonetheless, this approximation turns out to work in practice, and the implementation of the update rule is relatively simple.

The major stumbling block if we want to apply (72) to large problems is that the update rule requires the inversion of an $m \times m$ matrix. This is costly, and effectively precludes efficient exact solutions for problems of size significantly larger than $10^3$, due to memory and computational requirements. If we are able to provide a low rank approximation of $K$ by

$$\tilde{K} = U^\top K_{\text{sub}} U \text{ where } U \in \mathbb{R}^{n \times m} \text{ and } K_{\text{sub}} \in \mathbb{R}^{n \times n} \tag{73}$$

with $n \ll m$, however, we may compute (72) much more efficiently. For instance, it follows immediately from the Sherman-Woodbury-Morrison formula [22],

$$(V + RHR^\top)^{-1} = V^{-1} - V^{-1}R(H^{-1} + R^\top V^{-1} R)^{-1} R^\top V^{-1}, \tag{74}$$

that we obtain the following update rule for $\tilde{K}$,

$$\boldsymbol{\alpha}_{\text{new}} = \left(\mathbf{1} - U^\top \left(K_{\text{sub}}^{-1} + UCU^\top\right)^{-1} UC\right)(U^\top K_{\text{sub}} UC\boldsymbol{\alpha}_{\text{old}} - c). \tag{75}$$

[9] Strictly speaking, the Laplace approximation refers only to the fact that we approximate the mode of the posterior by a Gaussian distribution. We already use the Gaussian approximation in the second order method, however, in order to maximize the posterior. Hence, for all practical purposes, the two approximations just represent two different points of view on the same subject.

In particular, the number of operations required to solve (72) is $O(mn^2 + n^3)$ rather than $O(m^3)$. Numerically more stable, yet efficient and easily implementable methods than the Sherman-Woodbury-Morrison method exist, however their discussion would be somewhat technical. See [69, 12, 21] for further details and references.

There are several ways to obtain a good approximation of (73). One way is to project $k(x_i, x)$ on a random subset of dimensions, and express the missing terms as a linear combination of the resulting sub-matrix (this is the Nyström method proposed by Seeger and Williams [83]). We might also construct a randomized sparse greedy algorithm to select the dimensions (see [68] for details), or resort to a positive diagonal pivoting strategy [12].

An approximation of $K$ by its leading principal components, as often done in machine learning, is usually undesirable, since the computation of the eigensystem would still be costly, and the time required for prediction would still rise with the number of observations (since we cannot expect the leading eigenvectors of $K$ to contain a significant number of zero coefficients).

## 4.2 Variational Methods

In the case of logistic regression, Jaakkola and Jordan [32] compute upper and lower bounds on the logistic $(1 + e^{-t})^{-1}$, by exploiting the log-concavity of eq:gp:logistic-model: A convex function can be bounded from below by its tangent at any point, and from above by a quadratic with sufficiently large curvature (provided the maximum curvature of the original function is bounded). These bounds are

$$p(y = 1|t) \geq \frac{1}{1 + e^{-\nu}} \exp\left(\frac{(t - \nu)}{2} - \lambda(\nu)(t^2 - \nu^2)\right), \tag{76}$$

$$p(y = 1|t) \leq \exp\left(\mu t - H(\mu)\right), \tag{77}$$

where $\mu, \nu \in [0, 1]$ and $\lambda(\nu) = \frac{(1+e^{-\nu})^{-1} - 1/2}{2\nu}$. Furthermore, $H(\mu)$ is the binary entropy function, $H(\mu) = -\mu \ln \mu - (1 - \mu) \ln(1 - \mu)$.

Likewise, bounds for $p(y = -1|t)$ follow from $p(y = -1|t) = 1 - p(y = 1|t)$. Equations (77) and (76) can be calculated quite easily, since they are linear or quadratic functions in $t$. This means that for *fixed* parameters $\mu$ and $\nu$, we can optimize an upper and a lower bound on the log posterior using the same techniques as in Gaussian process regression (Section 3).

Approximations (77) and (76) are only tight, however, if $\nu, \mu$ are chosen suitably. Therefore we have to adapt these parameters at every iteration (or after each exact solution), for instance by gradient descent (by minimizing the upper bound and maximizing the lower bound correspondingly). See [16, 17] for details. Again, factorizations for rank-degenerate matrices as in the previous section can be used for efficient implementation.

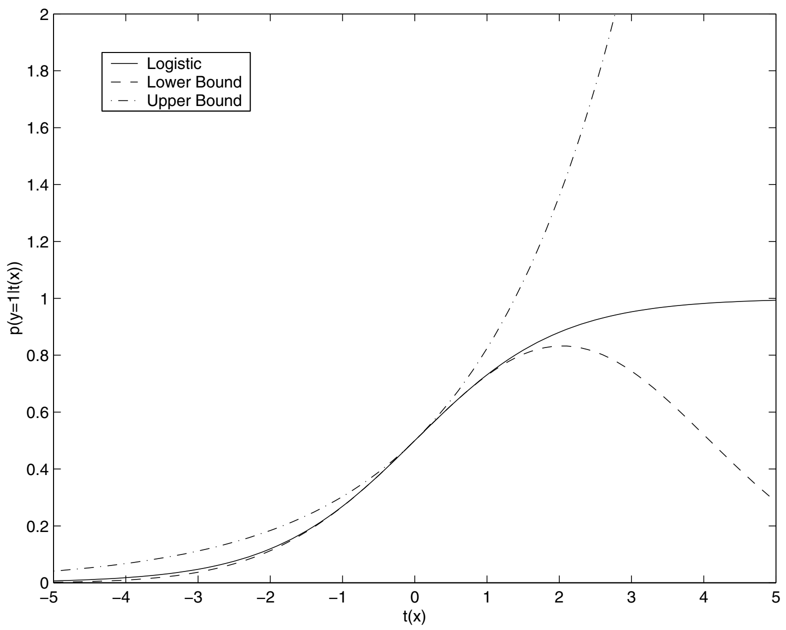

**Fig. 8.** Variational Approximation for $\mu = \nu = 0.5$. Note that the quality of the approximation varies widely, depending on the value of $f(x)$.

## 4.3 Approximate Solutions for Gaussian Process Regression

The approximations of Section 4.1 indicate that one of the more efficient ways of implementing Gaussian process estimation on large amounts of data is to find a low rank approximation[10] of the matrix $K$. Such an approximation is very much needed in practice, since (47) and (48) show that exact solutions of Gaussian Processes can be hard to come by. Even if $\boldsymbol{\alpha}$ is computed beforehand (see Table 1 for the scaling behavior), prediction of the mean at a new location still requires $O(m)$ operations. In particular, memory requirements are $O(m^2)$ to store $K$, and CPU time for matrix inversions, as are typically required for second order methods, scales with $O(m^3)$.

Let us limit ourselves to an approximation of the MAP solution. One of the criteria to impose is that the posterior probability at the approximate solution be close to the maximum of the posterior probability. Note that this requirement is different from the requirement of closeness in the approximation itself, as represented for instance by the expansion coefficients (the latter requirement

[10] Tresp [74] devised an efficient way of estimating $f(x)$ if the test set is known at the time of training. He proceeds by projecting the estimators on the subspace spanned by the functions $k(\tilde{x}_i, \cdot)$, where $\tilde{x}_i$ are the training data. Likewise, Csató and Opper [9] design an iterative algorithm that performs gradient descent on partial posterior distributions and simultaneously projects the estimates onto a subspace.

was used by Gibbs and Mackay [18]). Proximity in the coefficients, however, is not what we want, since it does not take into account the importance of the individual variables. For instance, it is not invariant under transformations of scale in the parameters.

For the remainder of the current section, we consider only additive normal noise. Here, the log posterior takes a quadratic form, given by (67). The following theorem, which uses an idea from [18], gives a bound on the approximation quality of minima of quadratic forms and is thus applicable to (67). For convenience we rewrite (67) in terms of $\boldsymbol{\theta} = K\alpha$.

**Theorem 2 (Approximation Bounds for Quadratic Forms [67]).** *Denote by $K \in \mathbb{R}^{m \times m}$ a symmetric positive definite matrix, $\mathbf{y}, \boldsymbol{\alpha} \in \mathbb{R}^m$, and define the two quadratic forms*

$$\mathfrak{L}(\boldsymbol{\alpha}) := -\mathbf{y}^\top K\boldsymbol{\alpha} + \frac{1}{2}\boldsymbol{\alpha}^\top(\sigma^2 K + K^\top K)\boldsymbol{\alpha}, \tag{78}$$

$$\mathfrak{L}^*(\boldsymbol{\alpha}) := -\mathbf{y}^\top \boldsymbol{\alpha} + \frac{1}{2}\boldsymbol{\alpha}^\top(\sigma^2 \mathbf{1} + K)\boldsymbol{\alpha}. \tag{79}$$

*Suppose $\mathfrak{L}$ and $\mathfrak{L}^*$ have minima $\mathfrak{L}_{\min}$ and $\mathfrak{L}^*_{\min}$. Then for all $\boldsymbol{\alpha}, \boldsymbol{\alpha}^* \in \mathbb{R}^m$ we have*

$$\mathfrak{L}(\boldsymbol{\alpha}) \geq \mathfrak{L}_{\min} \geq -\frac{1}{2}\|\mathbf{y}\|^2 - \sigma^2\mathfrak{L}^*(\boldsymbol{\alpha}^*), \tag{80}$$

$$\mathfrak{L}^*(\boldsymbol{\alpha}^*) \geq \mathfrak{L}^*_{\min} \geq \sigma^{-2}\Big(-\frac{1}{2}\|\mathbf{y}\|^2 - \mathfrak{L}(\boldsymbol{\alpha})\Big), \tag{81}$$

*with equalities throughout when $\mathfrak{L}(\boldsymbol{\alpha}) = \mathfrak{L}_{\min}$ and $\mathfrak{L}^*(\boldsymbol{\alpha}^*) = \mathfrak{L}^*_{\min}$.*

Hence, by minimizing $\mathfrak{L}^*$ in addition to $\mathfrak{L}$, we can bound $\mathfrak{L}$'s closeness to the optimum and vice versa.

*Proof.* The minimum of $\mathfrak{L}(\boldsymbol{\alpha})$ is obtained for $\boldsymbol{\alpha}_{\text{opt}} = (K + \sigma^2\mathbf{1})^{-1}\mathbf{y}$ (which also minimizes $\mathfrak{L}^*$),[11] hence

$$\mathfrak{L}_{\min} = -\frac{1}{2}\mathbf{y}^\top K(K + \sigma^2\mathbf{1})^{-1}\mathbf{y} \text{ and } \mathfrak{L}^*_{\min} = -\frac{1}{2}\mathbf{y}^\top(K + \sigma^2\mathbf{1})^{-1}\mathbf{y}. \tag{82}$$

This allows us to combine $\mathfrak{L}_{\min}$ and $\mathfrak{L}^*_{\min}$ to $\mathfrak{L}_{\min} + \sigma^2\mathfrak{L}^*_{\min} = -\frac{1}{2}\|\mathbf{y}\|^2$. Since by definition $\mathfrak{L}(\boldsymbol{\alpha}) \geq \mathfrak{L}_{\min}$ for all $\boldsymbol{\alpha}$ (and likewise $\mathfrak{L}^*(\boldsymbol{\alpha}^*) \geq \mathfrak{L}^*_{\min}$ for all $\boldsymbol{\alpha}^*$), we may solve $\mathfrak{L}_{\min} + \sigma^2\mathfrak{L}^*_{\min}$ for either $\mathfrak{L}$ or $\mathfrak{L}^*$ to obtain lower bounds for each of the two quantities. This proves (80) and (81).

Equation (80) is useful for computing an approximation to the MAP solution (the objective function is identical to $\mathfrak{L}(\boldsymbol{\alpha})$, ignoring constant terms independent of $\boldsymbol{\alpha}$), whereas (81) can be used to obtain error bars on the estimate. To see this,

[11] If $K$ does not have full rank, $\mathfrak{L}(\boldsymbol{\alpha})$ still attains its minimum value for $\boldsymbol{\alpha}_{\text{opt}}$. There will then be additional $\boldsymbol{\alpha}'$ that minimize $\mathfrak{L}(\boldsymbol{\alpha})$, however.

note that in calculating the variance (47), the expensive quantity to compute is $-\mathbf{k}^\top(K+\sigma^2\mathbf{1})^{-1}\mathbf{k}$. This can be found as

$$-\mathbf{k}^\top(K+\sigma^2\mathbf{1})^{-1}\mathbf{k} = 2\min_{\boldsymbol{\alpha}\in\mathbb{R}^m}\left[-\mathbf{k}^\top\boldsymbol{\alpha}+\tfrac{1}{2}\boldsymbol{\alpha}^\top\left(\sigma^2\mathbf{1}+K\right)\boldsymbol{\alpha}\right], \tag{83}$$

however. A close look reveals that the expression inside the parentheses is $\mathfrak{L}^*(\boldsymbol{\alpha})$ with $\mathbf{y}=\mathbf{k}$ (see (79)). Consequently, an approximate minimizer of (83) gives an *upper bound* on the error bars, and lower bounds can be obtained from (81). In practice, we use the relative discrepancy between the upper and lower bounds,

$$\operatorname{Gap}(\boldsymbol{\alpha},\boldsymbol{\alpha}^*) := \frac{2(\mathfrak{L}(\boldsymbol{\alpha})+\sigma^2\mathfrak{L}^*(\boldsymbol{\alpha}^*)+\frac{1}{2}\|\mathbf{y}\|^2)}{-\mathfrak{L}(\boldsymbol{\alpha})+\sigma^2\mathfrak{L}^*(\boldsymbol{\alpha}^*)+\frac{1}{2}\|\mathbf{y}\|^2}, \tag{84}$$

to determine how much further the approximation has to proceed.

### 4.4 Solutions on Subspaces

The central idea of the algorithm below is that improvements in speed can be achieved by a reduction in the number of free variables. Denote by $P\in\mathbb{R}^{m\times n}$ with $m\geq n$ and $m,n\in\mathbb{N}$ an extension matrix (in other words, $P^\top$ is a projection), with $P^\top P=\mathbf{1}$. We make the ansatz

$$\boldsymbol{\alpha}_P := P\boldsymbol{\beta} \text{ where } \boldsymbol{\beta}\in\mathbb{R}^n, \tag{85}$$

and find solutions $\boldsymbol{\beta}$ such that $\mathfrak{L}(\boldsymbol{\alpha}_P)$ (or $\mathfrak{L}^*(\boldsymbol{\alpha}_P)$) is minimized. The solution is

$$\boldsymbol{\beta}_{\text{opt}} = \left(P^\top\left(\sigma^2K+K^\top K\right)P\right)^{-1}P^\top K^\top\mathbf{y}. \tag{86}$$

If $P$ is of rank $m$, then $K\boldsymbol{\alpha}_P$ is also the minimizer of (67). In all other cases, however, it is an approximation.

For a given $P\in\mathbb{R}^{m\times n}$, let us analyze the computational cost involved in computing (86). We need $O(nm)$ operations to evaluate $P^\top K\mathbf{y}$, $O(n^2m)$ operations for $(KP)^\top(KP)$, and $O(n^3)$ operations for the inversion of an $n\times n$ matrix. This brings the total cost to $O(n^2m)$. Predictions require $\mathbf{k}^\top\boldsymbol{\alpha}$, which entails $O(n)$ operations. Likewise, we may use $P$ to minimize $\mathfrak{L}^*(P\boldsymbol{\beta}^*)$, which is needed to upper-bound the log posterior. The latter costs no more than $O(n^3)$.

To compute the posterior variance, we have to approximately minimize (83), which can done for $\boldsymbol{\alpha}=P\boldsymbol{\beta}$ at cost $O(n^3)$ . If we compute $(PKP^\top)^{-1}$ beforehand, the cost becomes $O(n^2)$, and likewise for upper bounds. In addition to this, we have to minimize $-\mathbf{k}^\top KP\boldsymbol{\beta}+\frac{1}{2}\boldsymbol{\beta}^\top P^\top(\sigma^2K+K^\top K)P\boldsymbol{\beta}$, which again costs $O(n^2m)$ (once the inverse matrices have been computed, however, we may also use them to compute error bars at different locations, thus limiting the cost to $O(n^2)$). Accurate lower bounds on the error bars are not especially crucial, since a bad estimate leads at worst to overly conservative confidence intervals, and has no further negative effect. Finally, note that we need only compute and store $KP$ — that is, the $m\times n$ sub-matrix of $K$ — and not $K$ itself. Table 1 summarizes the scaling behavior of several optimization algorithms.

**Table 1.** Computational Cost of Various Optimization Methods. Note that $n \ll m$, and that different values of $n$ are used in Conjugate Gradient, Sparse Decomposition, and Sparse Greedy Approximation methods: $n_{\mathrm{CG}} \leq n_{\mathrm{SD}} \leq n_{\mathrm{SGA}}$, since the search spaces are progressively more restricted. Near-optimal results are obtained when $\kappa = 60$.

| | Exact Solution | Conjugate Gradient [18] | Sparse Decomposition | Sparse Greedy Approximation |
|---|---|---|---|---|
| Memory | $O(m^2)$ | $O(m^2)$ | $O(nm)$ | $O(nm)$ |
| Initialization (= Training) | $O(m^3)$ | $O(nm^2)$ | $O(n^2m)$ | $O(\kappa n^2 m)$ |
| Prediction: | | | | |
| Mean | $O(m)$ | $O(m)$ | $O(n)$ | $O(n)$ |
| Error Bars | $O(m^2)$ | $O(nm^2)$ | $O(n^2m)$ or $O(n^2)$ | $O(\kappa n^2 m)$ or $O(n^2)$ |

This leads us to the question of how to choose $P$ for optimum efficiency. Possibilities include using the principal components of $K$ [83], performing conjugate gradient descent to minimize $\mathfrak{L}$ [18], performing symmetric diagonal pivoting [12], or applying a sparse greedy approximation to $K$ directly [68]. Yet these methods have the disadvantage that they either do not take the specific form of $\mathbf{y}$ into account [83, 68, 12], or lead to expansions that cost $O(m)$ for prediction, and require computation *and storage* of the full matrix [83, 18].

By contrast to these methods, we use a *data adaptive* version of a sparse greedy approximation algorithm. We may then only consider matrices $P$ that are a collection of unit vectors $\mathbf{e}_i$ (here $(\mathbf{e}_i)_j = \delta_{ij}$), since these only select a number of rows of $K$ equal to the rank of $P$ (see also [62] for a template of a sparse greedy algorithm).

- First, for $n = 1$, we choose $P = e_i$ such that $\mathfrak{L}(P\boldsymbol{\beta})$ is minimal. In this case we could permit ourselves to consider *all possible* indices $i \in \{1, \dots m\}$ and find the best one by trying all of them.
- Next, assume that we found a good solution $P\boldsymbol{\beta}$, where $P$ contains $n$ columns. In order to improve this solution, we expand the projection operator $P$ into the matrix $P_{\mathrm{new}} := [P_{\mathrm{old}}, \mathbf{e}_i] \in \mathbb{R}^{m \times (n+1)}$ and seek the best $\mathbf{e}_i$ such that $P_{\mathrm{new}}$ minimizes $\min_{\boldsymbol{\beta}} \mathfrak{L}(P_{\mathrm{new}}\boldsymbol{\beta})$.

Note that this method is very similar to Matching Pursuit [42] and to iterative reduced set Support Vector algorithms [60], with the difference that the target to be approximated (the full solution $\boldsymbol{\alpha}$) is only given implicitly via $\mathfrak{L}(\boldsymbol{\alpha})$.

Recently Zhang [85] proved lower bounds on the rate of sparse approximation schemes. In particular, he shows that most subspace projection algorithms enjoy at least an $O(n^{-1})$ rate of convergence.

## 4.5 Implementation Issues

Performing a full search over all possible $n + 1$ of $m$ indices is excessively costly. Even a full search over all $m - n$ remaining indices to select the next basis function can be prohibitively expensive. A simple result concerning the tails of rank statistics (see [62]) comes to our aid — it states that with high probability, a small subset of size $\kappa = 59$, chosen at random, guarantees near optimal performance. Hence, if we are satisfied with finding a *relatively good* index rather than the *best* index, we may resort to selecting only a random subset.

**Algorithm 1.1** Sparse Greedy Quadratic Minimization.

```
Require: Training data $X = \{x_1, \dots, x_m\}$, Targets $\mathbf{y}$, Noise $\sigma^2$, Precision $\epsilon$, corresponding quadratic forms $\mathfrak{L}$ and $\mathfrak{L}^*$.
  Initialize index sets $I, I^* = \{1, \dots, m\}$; $S, S^* = \emptyset$.
  repeat
    Choose $M \subseteq I$, $M^* \subseteq I^*$.
    Find $\operatorname{argmin}_{i \in M} Q\left([P, e_i]\boldsymbol{\beta}_{\text{opt}}\right)$, $\arg\min_{i^* \in M^*} \mathfrak{L}^*\left([P^*, e_{i^*}]\boldsymbol{\beta}^*_{\text{opt}}\right)$.
    Move $i$ from $I$ to $S$, $i^*$ from $I^*$ to $S^*$.
    Set $P := [P, e_i]$, $P^* := [P^*, e_{i^*}]$.
  until $\mathfrak{L}(P\boldsymbol{\beta}_{\text{opt}}) + \sigma^2\mathfrak{L}^*(P\boldsymbol{\beta}^*_{\text{opt}}) + \frac{1}{2}\|\mathbf{y}\|^2 \le \frac{\epsilon}{2}(|\mathfrak{L}(P\boldsymbol{\beta}_{\text{opt}})| + |\sigma^2\mathfrak{L}^*(P\boldsymbol{\beta}^*_{\text{opt}}) + \frac{1}{2}\|\mathbf{y}\|^2|$
Output: Set of indices $S$, $\boldsymbol{\beta}_{\text{opt}}$, $(P^\top K P)^{-1}$, and $(P^\top(K^\top K + \sigma^2 K)P)^{-1}$.
```

It is now crucial to obtain the values of $\mathfrak{L}(P\boldsymbol{\beta}_{\text{opt}})$ cheaply (with $P = [P_{\text{old}}, \mathbf{e}_i]$), assuming that we found $P_{\text{old}}$ previously. From (86) we can see that we need only do a rank-1 update on the inverse. We now show that this can be obtained in $O(mn)$ operations, provided the inverse of the smaller subsystem is known. Expressing the relevant terms using $P_{\text{old}}$ and $\mathbf{k}_i$, we obtain

$$P^\top K^\top \mathbf{y} = [P_{\text{old}}, \mathbf{e}_i]^\top K^\top \mathbf{y} = (P_{\text{old}}^\top K^\top \mathbf{y}, \mathbf{k}_i^\top \mathbf{y}), \tag{87}$$

$$P^\top \left(K^\top K + \sigma^2 K\right) P = \begin{bmatrix} P_{\text{old}}^\top \left(K^\top K + \sigma^2 K\right) P_{\text{old}} & P_{\text{old}}^\top \left(K^\top + \sigma^2 \mathbf{1}\right) \mathbf{k}_i \\ \mathbf{k}_i^\top (K + \sigma^2 \mathbf{1}) P_{\text{old}} & \mathbf{k}_i^\top \mathbf{k}_i + \sigma^2 K_{ii} \end{bmatrix} \tag{88}$$

Thus computation of the terms costs only $O(nm)$ once we know $P_{\text{old}}$. Furthermore, we can write the inverse of a strictly positive definite matrix as

$$\begin{bmatrix} A & B \\ B^\top & C \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} + (A^{-1}B)^\top \gamma (A^{-1}B) & -\gamma(A^{-1}B) \\ -(\gamma(A^{-1}B))^\top & \gamma \end{bmatrix}, \tag{89}$$

where $\gamma := (C - B^\top A^{-1} B)^{-1}$. Hence, inversion of $P^\top \left(K^\top K + \sigma^2 K\right) P$ costs only $O(n^2)$. Thus, to find the matrix $P$ of size $m \times n$ takes $O(\kappa n^2 m)$ time. For the error bars, $(P^\top K P)^{-1}$ is generally a good starting value for the minimization of (83), so the typical cost for (83) is $O(\tau m n)$ for some $\tau < n$, rather than $O(mn^2)$.

If additional numerical stability is required, we might want to replace (89) by a rank-1 update rule for Cholesky decompositions of the corresponding positive definite matrix. Furthermore, we may want to add the kernel function chosen by positive diagonal pivoting [12] to the selected subset, in order to ensure that the $n \times n$ sub-matrix remains invertible. See numerical mathematics textbooks, such as [28], for more detail on update rules.

## 4.6 Hardness and Approximation Results

It is worthwhile to study the theoretical guarantees on the performance of the algorithm (as described in Algorithm 1.1). It turns out that our technique closely resembles a Sparse Linear Approximation problem studied by Natarajan [44]:

*Given $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $\epsilon > 0$, find $x \in \mathbb{R}^n$ with minimal number of nonzero entries such that $\|Ax - b\|_2 \leq \epsilon$.* If we define

$$A = \left(\sigma^2 K + K^\top K\right)^{\frac{1}{2}} \text{ and } b := A^{-1} K \mathbf{y}, \tag{90}$$

we may write

$$\mathcal{L}(\boldsymbol{\alpha}) = \frac{1}{2}\|b - A\boldsymbol{\alpha}\|^2 + c, \tag{91}$$

where $c$ is a constant independent of $\boldsymbol{\alpha}$. Thus the problem of sparse approximate minimization of $\mathcal{L}(\boldsymbol{\alpha})$ is a special case of Natarajan's problem (where the matrix $A$ is square, and strictly positive definite). In addition, the algorithm considered in [44] involves sequentially choosing columns of $A$ to maximally decrease $\|Ax - b\|$. This is equivalent to the algorithm described above and we may apply the following result to our sparse greedy Gaussian process algorithm.

**Theorem 3 (Natarajan, 1995 [44]).** *A sparse greedy algorithm to approximately solve the problem*

$$\textit{minimize } \|y - Ax\| \tag{92}$$

*needs at most*

$$n \leq 18 n^*(\epsilon) \|A^+\|_2^2 \ln \frac{\|y\|}{2\epsilon} \tag{93}$$

*non-zero components, where $n^*(\epsilon)$ is the minimum number of nonzero components in vectors $\boldsymbol{\alpha}$ for which $\|y - Ax\| \leq \epsilon$, and $A^+$ is the matrix obtained from $A$ by normalizing its columns to unit length.*

**Corollary 1 (Approximation Rate for Gaussian Processes).** *Algorithm 1.1 satisfies $\mathcal{L}(\boldsymbol{\alpha}) \leq \mathcal{L}(\boldsymbol{\alpha}_{\text{opt}}) + \epsilon^2$ when $\boldsymbol{\alpha}$ has*

$$n \leq \frac{18 n^*(\epsilon)}{\lambda^2} \ln \frac{\|A^{-1} K y\|}{2\epsilon} \tag{94}$$

*non-zero components, where $n^*(\epsilon)$ is the minimum number of nonzero components in vectors $\boldsymbol{\alpha}$ for which $\mathcal{L}(\boldsymbol{\alpha}) \leq \mathcal{L}(\boldsymbol{\alpha}_{\text{opt}}) + \epsilon^2$, $A = (\sigma^2 K + K^\top K)^{1/2}$, and $\lambda$ is the smallest magnitude of the singular values of $\mathbf{A}$, the matrix obtained by normalizing the columns of $A$.*

Moreover, we can also show NP-hardness of sparse approximation of Gaussian process regression. The following theorem holds:

**Theorem 4 (NP-Hardness of Approximate GP Regression).** *There exist kernels $K$ and labels $\mathbf{y}$ such that the problem of finding the minimal set of indices to minimize a corresponding quadratic function $\mathfrak{L}(\boldsymbol{\alpha})$ with precision $\varepsilon$ is NP-hard.*

*Proof.* We use the hardness result [44, Theorem 1] for Natarajan's quadratic approximation problem in terms of $A$ and $b$. More specifically, we have to proceed in the opposite direction to (90) and (91) and show that for every $A$ and $b$, there exist $K$ and $\mathbf{y}$ for an equivalent optimization problem.

Since $\|Ax - b\|^2 = x^\top(A^\top A)x - 2(b^\top A)x + \|b\|^2$, the value of $A$ enters only via $A^\top A$, which means that we have to find $K$ in (78) such that

$$A^\top A = K^\top K + \sigma^2 K. \tag{95}$$

We can check that it is possible to find a suitable positive definite $K$ for any $A$, by using identical eigensystems for $A^\top A$ and $K$, and subsequently solving the equations $a_i = \lambda_i^2 + \sigma^2\lambda_i$ for the respective eigenvalues $a_i$ and $\lambda_i$ of $A^\top A$ and $K$. Furthermore, we have to satisfy

$$\mathbf{y}^\top K = bA. \tag{96}$$

To see this, recall that $bA$ is a linear combination of the nonzero eigenvectors of $A^\top A$; and since $K$ has the same rank and image as $A^\top A$, the vector $bA$ can also be represented by $\mathbf{y}^\top K$. Thus for every $A, b$ there exists an equivalent $\mathfrak{L}$, which proves NP-hardness by reduction.

This shows that the sparse greedy algorithm is an efficient approximate solution of an NP-hard problem.

## 4.7 Experimental Evidence

We conclude this section with a brief experimental demonstration of the efficiency of sparse greedy approximation methods, using the Abalone dataset. Specifically, we used Gaussian covariance kernels, and we split the data into 4000 training and 177 test examples to assess training speed (to assess generalization performance, a 3000 training and 1177 test set split was used).

For the optimal parameters ($2\sigma^2 = 0.1$ and $2\omega^2 = 10$, chosen after [68]), the average test error of the sparse greedy approximation trained until $\mathrm{Gap}(\boldsymbol{\alpha}, \boldsymbol{\alpha}^*) < 0.025$ is indistinguishable from the corresponding error obtained by an exact solution of the full system. The same applies for the log posterior. See Table 2 for details. Consequently for all practical purposes, full inversion of the covariance matrix and the sparse greedy approximation have comparable generalization performance.

A more important quantity in practice is the number basis functions needed to minimize the log posterior to a sufficiently high precision. Table 3 shows this number for a precision of $\mathrm{Gap}(\boldsymbol{\alpha}, \boldsymbol{\alpha}^*) < 0.025$, and its variation as a function of the kernel width $\sigma$; the latter dependency is observed since the number of

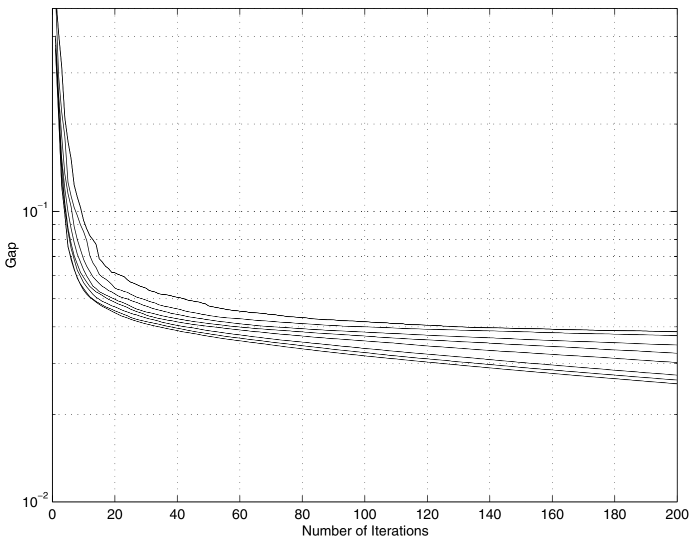

**Fig. 9.** Speed of Convergence. We plot the size of the gap between upper and lower bound of the log posterior (Gap($\boldsymbol{\alpha}, \boldsymbol{\alpha}^*$)), for the first 4000 samples from the Abalone dataset ($\sigma^2 = 0.1$ and $2\omega^2 = 10$). From top to bottom: Subsets of size 1, 2, 5, 10, 20, 50, 100, 200. The results were averaged over 10 runs. The relative variance of the gap size is less than 10%. We can see that subsets of size 50 and above ensure rapid convergence.

kernels determines time and memory needed for prediction and training. In all cases, less than 10% of the kernel functions suffice to find a good minimizer of the log posterior; less than 2% are sufficient to compute the error bars. This is a significant improvement over a direct minimization approach.

A similar result can be obtained on larger datasets. To illustrate, we generated a synthetic data set of size 10.000 in $\mathbb{R}^{20}$ by adding normal noise with variance $\sigma^2 = 0.1$ to a function consisting of 200 randomly chosen Gaussians of width $2\omega^2 = 40$ and with normally distributed expansion coefficients and centers.

To avoid trivial sparse expansions, we deliberately used an *inadequate* Gaussian process prior (but correct noise level) consisting of Gaussians with width $2\sigma^2 = 10$. After 500 iterations (thus, after using 5% of all basis functions), the size of Gap($\boldsymbol{\alpha}, \boldsymbol{\alpha}^*$) was less than 0.023. This demonstrates the feasibility of the sparse greedy approach on larger datasets.

## 5 Laplacian Processes

So far the dependency of the latent variables $\boldsymbol{\theta}$ on $X$ could not be factorized easily in terms of the $x_i$. That is, $p(\boldsymbol{\theta}|X) \neq \prod_i p(\theta_i|x_i)$. This is due to the fact

**Table 2.** Performance of sparse greedy approximation vs. explicit solution of the full learning problem. In these experiments, the Abalone dataset was split into 3000 training and 1177 test samples. To obtain more reliable estimates, the algorithm was run over 10 random splits of the whole dataset.

| | Generalization Error | Log Posterior |
|---|---|---|
| Optimal Solution | $1.782 \pm 0.33$ | $-1.571 \cdot 10^5(1 \pm 0.005)$ |
| Sparse Greedy Approximation | $1.785 \pm 0.32$ | $-1.572 \cdot 10^5(1 \pm 0.005)$ |

**Table 3.** Number of basis functions needed to minimize the log posterior on the Abalone dataset (4000 training samples), for various kernel widths $\omega$. Also given is the number of basis functions required to approximate $\mathbf{k}^\top(K + \sigma^2\mathbf{1})^{-1}\mathbf{k}$, which is needed to compute the error bars. Results were averaged over the 177 test samples.

| Kernel width | 1 | 2 | 5 | 10 | 20 | 50 |
|---|---|---|---|---|---|---|
| Kernels for log-posterior | 373 | 287 | 255 | 257 | 251 | 270 |
| Kernels for error bars | 79±61 | 49±43 | 26±27 | 17±16 | 12±9 | 8±5 |

that we want to infer from knowing $(x, y)$ the likely value of $(x', y')$ — a model which did not *couple* the various $y_i$ would fail in this regard.

However, by a simple trick, we can achieve the factorization, yet still retain the inference properties of the estimator: we introduce yet another layer of dependence: rather than modelling

$$\xrightarrow{p(X)} X \xrightarrow[p(\boldsymbol{\theta}|X)p(X)]{p(\boldsymbol{\theta}|X)} \boldsymbol{\theta} \xrightarrow[p(Y|\boldsymbol{\theta})p(\boldsymbol{\theta}|X)p(X)]{p(Y|\boldsymbol{\theta})} Y$$

we assume

$$\xrightarrow{p(X)} X \xrightarrow[p(\boldsymbol{\theta}|X)p(X)]{p(\boldsymbol{\theta}|X)} \boldsymbol{\theta} \xrightarrow[p(\boldsymbol{\theta}|X)p(X)]{\mathbf{t}=K\boldsymbol{\theta}} \mathbf{t} \xrightarrow[p(Y|\mathbf{t}(\boldsymbol{\theta}))p(\boldsymbol{\theta}|X)p(X)]{p(Y|\mathbf{t})} Y$$

where $p(\boldsymbol{\theta}|X) = \prod_{i=1}^m p(\theta_i|x_i)$ and $p(Y|\mathbf{t}(\boldsymbol{\theta})) = \prod_{i=1}^m p(y_i|t_i(\boldsymbol{\theta}))$. That is, we moved the mixing between the various $y_i$ into the design matrix $K$. Note that there is no requirement that $K$ be positive semidefinite. In fact, any arbitrary matrix would do. However, for practical purposes we will typically choose some function $k$ with $K_{ij} = k(x_i, x_j)$.

Before we go into the technical details, let us give some motivation as to why the complexity of an estimate can depend on the locations where data occurs, since we are effectively updating our prior assumptions about $\mathbf{t}$ after observing the data placement. Note that we do not modify our prior assumptions based on the targets $y_i$, but rather as a result of the distribution of patterns $x_i$: Different input distribution densities might for instance correspond to different assumptions regarding the smoothness of the function class to be estimated. For example, it might be be advisable to favor smooth functions in areas where data are scarce, and allow more complicated functions where observations abound. We might not care about smoothness at all in regions where there is little or no chance of patterns occurring: In the problem of handwritten digit recognition, we do not (and should not) care about the behavior of the estimator on inputs $x$ looking like faces.

The specific benefit of this strategy is that it provides us with a correspondence between linear programming regularization [43, 2, 65, 6] and Bayesian priors over function spaces, by analogy to regularization in Reproducing Kernel Hilbert Spaces and Gaussian Processes.[12]

## 5.1 Examples of Factorizing Priors

Let us now study some of the priors factorizing in coefficient space. By construction we have

$$p(\mathbf{t}|X) = \frac{1}{\mathcal{Z}} \exp\left(-\sum_{i=1}^{m} \gamma(\theta_i)\right), \text{ where } \mathbf{t}_i = \sum_{i=1}^{m} \theta_i k(x_i, x). \tag{97}$$

Here $\gamma(\theta_i)$ is chosen such that $\exp(-\gamma(\theta))$ is integrable, $\mathcal{Z}$ is the corresponding normalization term, and $x_i \in X$. Examples of priors that depend on the locations $x_i$ include

$$\gamma(\theta) = 1 - e^{p|\theta|} \text{ with } p > 0 \text{ (feature selection prior)}, \tag{98}$$

$$\gamma(\theta) = \theta^2 \text{ (weight decay prior)}, \tag{99}$$

$$\gamma(\theta) = |\theta| \text{ (Laplacian prior)}. \tag{100}$$

The prior given by (98) was introduced in [5, 14] and is log-concave. While the latter characteristic is unfavorable in general, since the corresponding optimization problem exhibits many local minima, the negative log-posterior becomes strictly concave if we choose Laplacian noise (equivalent to the $L_1$ loss in regression). By a basic result from convex analysis [57] this means that the optimum occurs at one of the extreme points, which makes optimization more feasible.

Eq. (99) describes the popular *weight decay* prior used in Bayesian Neural Networks [40, 45, 46]. It assumes that the coefficients are independently normally distributed. We relax the assumption of a common normal distribution in Section 6 and introduce individual (hyper)parameters $s_i$. The resulting prior,

$$p(\boldsymbol{\theta}|X, s) = (2\pi)^{-\frac{m}{2}} \left(\prod_{i=1}^{m} s_i\right)^{\frac{1}{2}} \exp\left(-\frac{1}{2}\sum_{i=1}^{m} s_i\theta_i^2\right), \tag{101}$$

leads to the construction of the Relevance Vector Machine [73] and very sparse function expansions.

Finally, the assumption underlying the Laplacian prior (100) is that only very few basis functions will be nonzero. The specific form of the prior is why we will call such estimators *Laplacian Processes.* This prior has two significant advantages over (98): It leads to convex optimization problems, and the integral

[12] We thank Carl Magnus Rasmussen for discussions and suggestions.

$\int p(\theta)d\theta$ is finite and thus allows normalization (this is not the case for (98), which is why we call the latter an *improper* prior).

The Laplacian prior corresponds to the regularization functional employed in sparse coding approaches, such as wavelet dictionaries [6], coding of natural images [48], independent component analysis [37], and linear programming regression [66, 65].

In the following, we focus on (100). It is straightforward to see that the MAP estimate can be obtained by minimizing the negative log posterior, which is given (up to constant terms) by

$$-\sum_{i=1}^{m} \ln p(y_i|f(x_i), x_i) + \sum_{i=1}^{m} |\theta_i|. \tag{102}$$

Depending on $\ln p(y_i|t_i(\boldsymbol{\theta}))$, we may formulate the minimization of (102) as a linear or quadratic program.

## 5.2 Samples from the Prior

In order to illustrate our reasoning, and to show that such priors correspond to useful distributions over $\mathbf{t}$, we generate samples from the prior distribution. As in Gaussian processes, smooth kernels $k$ correspond to smooth priors. This is not surprising: As we show in the next section (Theorem 5), there exists a corresponding Gaussian process for every kernel $k$ and every distribution $p(x)$.

The obvious advantage, however, is that we need not worry about Mercer's condition for $k$ but can take any arbitrary function $k(x, x')$ to generate a Laplacian process. We draw samples from the following three kernels,

$$k(x, x') = e^{-\frac{\|x-x'\|^2}{2\sigma^2}} \quad \text{Gaussian RBF kernel,} \tag{103}$$

$$k(x, x') = e^{-\frac{\|x-x'\|}{\sigma}} \quad \text{Laplacian RBF kernel,} \tag{104}$$

$$k(x, x') = \tanh(\theta\langle x, x'\rangle + \vartheta) \quad \text{Neural Networks kernel.} \tag{105}$$

While (103) and (104) are also valid kernels for Gaussian Process estimation, (105) does not satisfy Mercer's condition[13]. Figure 10 gives sample realizations from the corresponding process. The use of (105) is impossible for GP priors, unless we diagonalize the matrix $K$ explicitly and render it positive definite by replacing $\lambda_i$ with $|\lambda_i|$. This is a very costly procedure (see also [61, 24]) as it involves computing the eigensystem of $K$.

## 5.3 Estimation

Since one of the aims of using a Laplacian prior on the coefficients $\theta_i$ is to achieve sparsity of the expansion, it does not appear sensible to use a Bayesian averaging

[13] The covariance matrix $K$ has to be positive definite at all times. An analogous application of the theory of conditionally positive definite kernels would be possible as well. There one simply assumes a Gaussian Process prior on a linear subspace of the $y_i$.

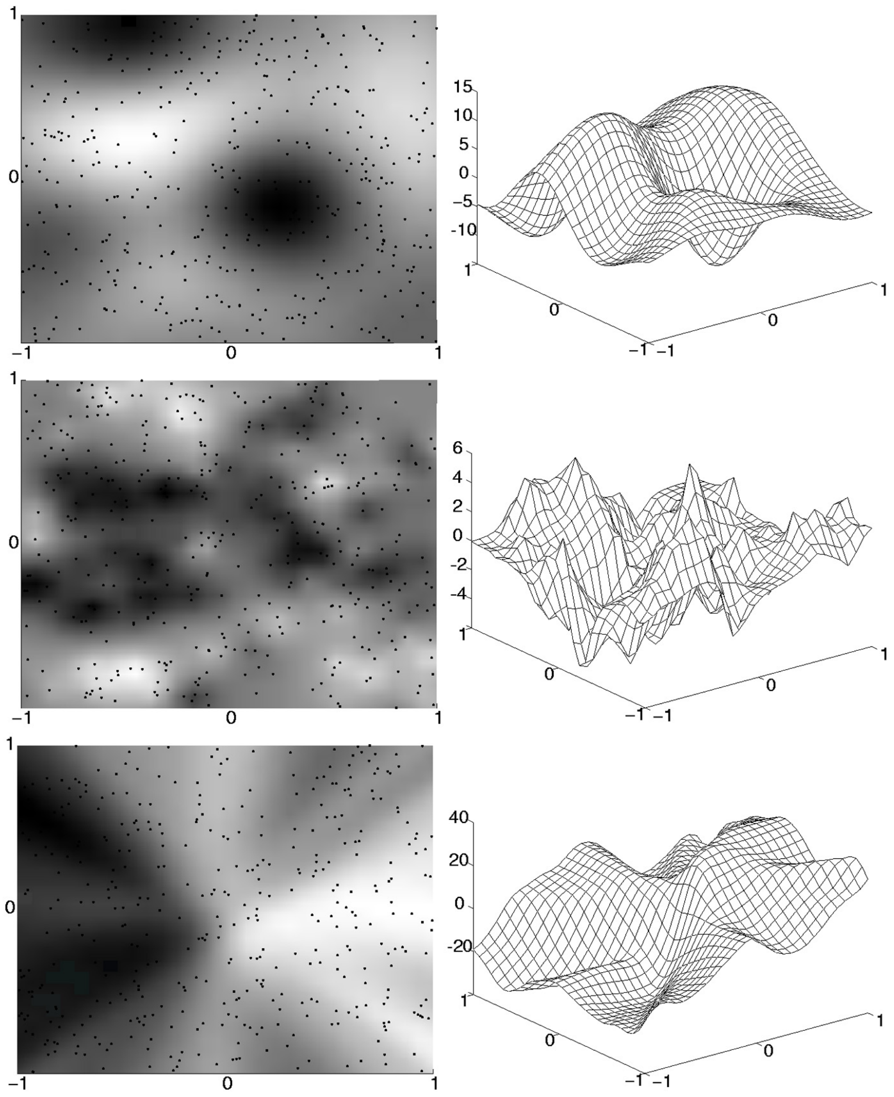

**Fig. 10.** Left Column: Grayscale plots of the realizations of several Laplacian Processes. The black dots represent data points. Right Column: 3D plots of the same samples of the process. We used 400 data points sampled at random from $[0,1]^2$ using a uniform distribution. Top to bottom: Gaussian kernel (103) ($\sigma^2 = 0.1$), Laplacian kernel (104) ($\sigma = 0.1$), and Neural Networks kernel (105) ($\theta = 10, \vartheta = 1$). Note that the Laplacian kernel is significantly less smooth than the Gaussian kernel, as with a Gaussian Process with Laplacian kernels. Moreover, observe that the Neural Networks kernel corresponds to a non-stationary process; that is, its covariance properties are not translation invariant.

scheme to compute the mean of the posterior distribution, since such a scheme leads to mostly nonzero coefficients. Instead we seek to obtain the *mode* of the distribution (the MAP estimate) only.

As already pointed out in the previous section, finding the mode need not give an exact solution, since mode and mean do not coincide for Laplacian regularization (recall Figure 3). Nonetheless, the MAP estimate is computationally attractive, since if both $-\ln p(\xi_i)$ and $\gamma(\theta_i)$ are convex, the optimization problem has a unique minimum.

By assumption we can write the joint density in $Y, \mathbf{t}, \boldsymbol{\theta}$ as follows:

$$p(Y, \mathbf{t}, \boldsymbol{\theta}|X) = p(Y|\mathbf{t}(\boldsymbol{\theta}))p(\boldsymbol{\theta}|X) \tag{106}$$

$$= \left[\prod_{i=1}^{m} p(y_i|t_i(\boldsymbol{\theta}))p(\theta_i)\right] \text{ where } \mathbf{t}(\theta) = K\theta \tag{107}$$

$$\propto \left[\prod_{i=1}^{m} p(y_i|t_i(\boldsymbol{\theta})) \exp\left(-\gamma(\theta_i)\right)\right] \tag{108}$$

Here we exploited the fact that the prior factorizes and that the data is generated iid. Maximization of (106) is equivalent to minimizing the negative log posterior which leads to

$$\begin{aligned} &\text{minimize} && \sum_{i=1}^{m} -\ln p(y_i|t_i) + \sum_{i=1}^{m} \gamma(\theta_i), \\ &\text{subject to} && \mathbf{t} = K\boldsymbol{\theta} \end{aligned} \tag{109}$$

For $-\log p(\xi_i) = |\xi_i| + c$ and $\gamma(\theta_i) = |\theta_i|$, this leads to a linear program and the solution can be readily used as a MAP estimate for Laplacian processes (a similar reasoning holds for soft margin loss functions). Likewise for Gaussian noise, we obtain a quadratic program with a simple objective function but a dense set of constraints, by analogy to Basis Pursuit [6].

## 5.4 Confidence Intervals for Gaussian Noise

One of the key advantages of Bayesian modeling is that we can obtain explicit confidence intervals for the predictions, provided the assumptions made regarding the priors and distribution are satisfied. Even for Gaussian noise, however, no explicit meaningful expansion using the MAP estimate $\boldsymbol{\alpha}_{\mathrm{MAP}}$ is possible, since $\gamma(\theta_i) = |\theta_i|$ is non-differentiable at 0 (otherwise we could make a quadratic approximation at $\theta_i = 0$). Nonetheless, a slight modification permits computationally efficient approximation of such error bounds.

The modification consists of dropping all variables $\theta_i$ for which $\theta_{\mathrm{MAP},i} = 0$ from the expansion (this renders the distribution *flatter* and thereby overestimates the error), and replacing all remaining variables by linear approximations (we replace $|\theta_i|$ by $\operatorname{sgn}(\theta_{\mathrm{MAP},i})\,\theta_i$).

In other words, we assume that variables with zero coefficients do not influence the expansion, and that the signs of the remaining variables do not change.

This is a sensible approximation since for large sample sizes, which Laplacian processes are designed to address, the posterior is strongly peaked around its mode. Thus the contribution of $-\log p(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_1 + c$ around $\boldsymbol{\theta}_{\mathrm{MAP}}$ can be considered to be approximately linear.

Denote by $\boldsymbol{\theta}_M$ the vector of nonzero variables, obtained by deleting all entries where $\theta_{\mathrm{MAP},i} = 0$, by $s$ the vector with elements $\pm 1$ such that $\|\boldsymbol{\theta}_{\mathrm{MAP}}\|_1 = s^\top \boldsymbol{\theta}_M$, and by $K_M$ the matrix generated from $K$ by removing the columns corresponding to $\theta_{\mathrm{MAP},i} = 0$. Then the posterior (now written in terms of $\boldsymbol{\theta}$ for convenience) can be approximated as

$$p(\boldsymbol{\theta}_M|X,Y) \approx \frac{1}{\mathcal{Z}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{m}(y_i - K_M\boldsymbol{\theta}_M)^2\right)\exp\left(-s^\top\boldsymbol{\theta}_M\right). \tag{110}$$

Collecting linear and quadratic terms, we see that

$$\boldsymbol{\theta}_M \sim \mathcal{N}(\boldsymbol{\theta}_{\mathrm{MAP}}, (K_M^\top K_M)^{-1}), \text{ where } \boldsymbol{\theta}_{\mathrm{MAP}} = (K_M^\top K_M)^{-1}(K_M^\top y + \sigma^2 s). \tag{111}$$

The equation for $\boldsymbol{\theta}_{\mathrm{MAP}}$ follows from the conditions on the optimal solution of the quadratic programming problem (109), or directly from maximizing (106) (after $s$ is fixed). Hence predictions at a new point $x$ are approximately normally distributed, with

$$y(x) = \mathcal{N}\left(\mathbf{k}_M^\top\boldsymbol{\theta}_M, \left(\sigma^2 + \mathbf{k}_M^\top\left(K_M^\top K_M\right)^{-1}\mathbf{k}_M\right)\right), \tag{112}$$

where $\mathbf{k}_M := (k(x_1,x),\dots,k(x_M,x))$ and only $x_i$ with nonzero $\theta_{\mathrm{MAP},i}$ are considered (thus $M \leq m$). The additional $\sigma^2$ stems from the fact that we have additive Gaussian noise of variance $\sigma^2$ in addition to the Laplacian process. Equation (112) is still expensive to compute, but it is much cheaper to invert $\Sigma_M^\top \Sigma_M$ than a dense square matrix $\Sigma$ (since $\boldsymbol{\theta}_{\mathrm{MAP}}$ may be very sparse). In addition, greedy approximation methods (as described for instance in Section 4.4) or column generation techniques [1] could be used to render the computation of (112) numerically more efficient.

## 5.5 An Equivalent Gaussian Process

We conclude this section with a proof that in the large sample size limit, there exists a Gaussian Process for each kernel expansion with a prior on the coefficients $\theta_i$. For the purpose of the proof, we have to slightly modify the normalization condition on $f$: That is, we assume

$$y_i(\boldsymbol{\theta}) = m^{-\frac{1}{2}}\sum_{i=1}^{m}\theta_i k(x_i,x), \tag{113}$$

where $\theta_i \sim \frac{1}{\mathcal{Z}}\exp(-\gamma(\boldsymbol{\theta}))$. For large sample size, i.e., $m \to \infty$, the following theorem holds.

**Theorem 5 (Convergence to Gaussian Process).** *Denote by $\theta_i$ independent random variables (we do not require identical distributions on $\theta_i$) with unit variance and zero mean. Furthermore, assume that there exists a distribution $p(x)$ on $\mathcal{X}$ with according to which a sample $\{x_1, \dots, x_m\}$ is drawn, and that $k(x, x')$ is bounded on $\mathcal{X} \times \mathcal{X}$. Then the random variable $y(x)$ given by (113) converges for $m \to \infty$ to a Gaussian process with zero mean and covariance function*

$$\tilde{k}(x, x') = \int_{\mathcal{X}} k(x, \bar{x}) k(x', \bar{x}) p(\bar{x}) d\bar{x}. \tag{114}$$

This means that instead of a Laplacian process prior, we could use any other factorizing prior on the expansion coefficients $\theta_i$ and in the limit still obtain an equivalent stochastic process.

*Proof.* To prove the first part, we need only check is that $y(x)$ and any linear combination $\sum_j y(x_j)$ (for arbitrary $x'_j \in \mathcal{X}$) converge to a normal distribution. By application of a theorem of Cramér [8], this is sufficient to prove that $y(x)$ is distributed according to a Gaussian Process.

The random variable $y(x)$ is a sum of $m$ independent random variables with bounded variance (since $k(x, x')$ is bounded on $\mathcal{X} \times \mathcal{X}$). Therefore in the limit $m \to \infty$, by virtue of the Central Limit Theorem (e.g., [8]), we have $y(x) \sim \mathcal{N}(0, \sigma^2(x))$ for some $\sigma^2(x) \in \mathbb{R}$. For arbitrary $x'_j \in \mathcal{X}$, linear combinations of $y(x'_j)$ also have Gaussian distributions since

$$\sum_{j=1}^{n} \beta_i y_i = m^{-\frac{1}{2}} \sum_{i=1}^{m} \theta_i \sum_{j=1}^{n} \beta_j k(x_i, x_j), \tag{115}$$

which allows the application of the Central Limit Theorem to the sum since the inner sum $\sum_{j=1}^{n} \beta_j k(x_i, x_j)$ is bounded for any $x_i$. It also implies $\sum_{j=1}^{n} \beta_j y_j \sim \mathcal{N}(0, \sigma^2)$ for $m \to \infty$ and some $\sigma^2 \in \mathbb{R}$, which proves that $y(x)$ is distributed according to a Gaussian Process.

To show (114), first note that $y(x)$ has zero mean. Thus the covariance function for finite $m$ can be found as expectation with respect to $\boldsymbol{\theta}$,

$$E_{\boldsymbol{\theta}}[y(x)y(x')] = E_{\boldsymbol{\theta}} \left[ \frac{1}{m} \sum_{i,j=1}^{m} \theta_i \theta_j k(x_i, x) k(x_j, x') \right] = \frac{1}{m} \sum_{i=1}^{m} k(x_i, x) k(x_j, x'), \tag{116}$$

since the $\theta_i$ are independent and have zero mean. This expression, however, converges to the Riemann integral over $\mathcal{X}$ with the density $p(x)$ as $m \to \infty$. Thus

$$E[y(x)y(x')] \underset{m \to \infty}{\longrightarrow} \int_{\mathcal{X}} k(x, \bar{x}) k(x', \bar{x}) p(\bar{x}) d\bar{x}, \tag{117}$$

which completes the proof.

## 6 Relevance Vector Machines and Deconvolution

One of the problems with the probabilistic model introduced in (97) is that it may lead to somewhat intractable optimization problems, in particular if the negative log posterior $\gamma(\theta)$ is not convex. However, such functions $\gamma$ may correspond to useful statistical assumptions on the distribution of coefficients in function expansions.

### 6.1 Turning Priors into Hyperpriors

Recently, Tipping [73] proposed a method to circumvent the numerical problems inherent in using a certain set of $\gamma(\theta)$ via deconvolution. Before presenting the specific choices [73] makes for the sake of sparse function expansions, we present the general principle, since it can be extended to priors on coefficients and likelihood terms alike, using ideas from [20]. The method works as follows: Assume we have a prior

$$p(\theta_i|s_i) = \sqrt{\frac{s_i}{2\pi}} \exp\left(-\frac{1}{2}s_i\theta_i^2\right), \tag{118}$$

that is $s_i$ plays the role of a hyperprior and $\theta_i \sim \mathcal{N}(0, s_i^{-1})$. Here, given $s_i$, we can use well studied methods for inference with a Gaussian prior to perform the required estimation steps. Quite often we will be able to perform *exact* inference, given the hyperparameters $s_i$. Key is the suitable choice of $p(s_i)$, since clearly

$$p(\theta_i) = \int p(\theta_i|s_i)p(s_i)ds_i = \int \sqrt{\frac{s_i}{2\pi}} \exp\left(-\frac{1}{2}s_i\theta_i^2\right) p(s_i)ds_i. \tag{119}$$

This means that given some $p(\theta_i)$ we may be able to find some $p(s_i)$ such that (119) holds. A hyperprior $p(s_i)$ with a large weight at $s_i = 0$ is desirable since this leads to a of the distribution of $\theta_i$ around 0. A parameter transformation in the integral $\beta = \frac{1}{2s_i^2}$ yields

$$p(\theta_i) = \int \exp(-\beta\theta_i)(8\pi)^{-\frac{1}{2}}\beta^{-1}p\left((2\beta)^{-\frac{1}{2}})\right) d\beta. \tag{120}$$

That is, $p(\theta_i)$ is the *Laplace Transform* of $(8\pi)^{-\frac{1}{2}}\beta^{-1}p\left((2\beta)^{-\frac{1}{2}})\right)$ and correspondingly $p(s_i)$ is given by the *inverse Laplace Transform* of $p(\theta_i)$ and suitable variable changes. This fact allows us to match up priors $p(\theta_i)$ and corresponding hyperpriors $p(s_i)$ in a fairly automatic fashion.

In particular, we assume a normal distribution over $\theta_i$ with adjustable variance. The latter is then determined with a hyperparameter that has its most likely value at 0; This prior is expressed analytically as

**Normal Hyperprior:** We begin with a normal distribution on $p(s_i)$, that is

$$p(s_i) = \frac{1}{2\pi\omega^2} \exp\left(-\frac{1}{2\omega^2}s_i^2\right). \tag{121}$$

Performing a Laplace transform leads to

$$p(\theta_i) = \frac{1}{2\pi\theta_i}\text{BesselK}_0\left(\frac{\sqrt{8}}{\theta_i}\right), \tag{122}$$

where BesselK is the modified Bessel function of the second kind [79]. See Figure 11 for more properties of this function.

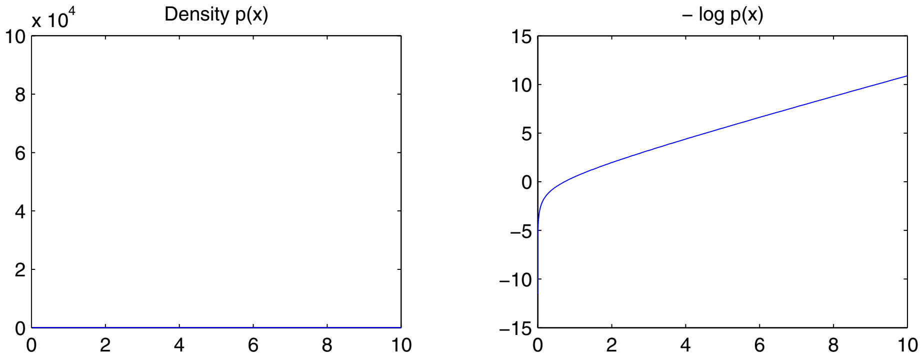

**Fig. 11.** $p(\theta_i)$ for a normal hyperprior. Left: $p(\theta_i)$; Right: $-\log p(\theta_i)$. Note the sharp peak at $\theta_i = 0$ and the almost linear increase afterwards.

**$\Gamma$-Hyperprior:** Tipping [73] used the Gamma hyperprior to design the Relevance Vector Machine. Here

$$p(s_i) = \Gamma(s_i|a,b) := \frac{s_i^{a-1}b^a\exp(-s_i b)}{\Gamma(a)} \text{ for } s_i > 0. \tag{123}$$

For non-informative (flat in logspace) priors, one typically chooses $a = b = 10^{-4}$. This leads to a polynomial prior which is given by

$$p(\theta_i) \propto \exp\left(-(a+1/2)\ln\left(b + \frac{\theta_i^2}{2}\right)\right) = \left(b + \frac{\theta_i^2}{2}\right)^{-(a+1/2)}. \tag{124}$$

Note that (123) is heavily peaked for $s_i \to 0$. For regression, a similar assumption is made concerning the amount of additive Gaussian noise $\sigma^2$; thus $p(\bar{\sigma}^2) = 1/\bar{\sigma}^2$ or $p(\bar{\sigma}^2) = \Gamma(\bar{\sigma}^2|c,d)$ where typically $c = d = 10^{-4}$. Note that the priors are imposed on the *inverse* of $\sigma$ and $\bar{\sigma}$. Figure 12 depicts the scaling behavior for non-informative priors.

Further choices are possible and can be obtained by consulting tables of Legendre transformations, such as in [23].

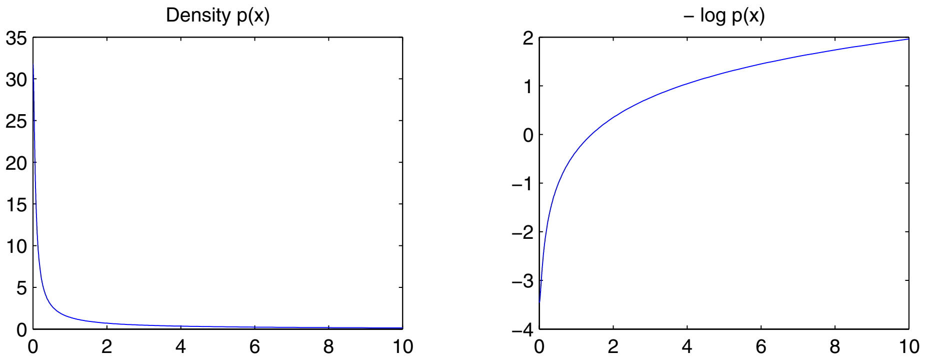

**Fig. 12.** $p(\theta_i)$ for a Gamma hyperprior ($a = b = 10^{-4}$). Left: $p(\theta_i)$; Right: $-\log p(\theta_i)$. Note that here the distribution is much less peaked around 0 and observe the sublinear increase in the negative log density.

## 6.2 Further Expansions

A similar approach can be used to transform arbitrary $p(\theta_i)$ into Laplacian distributions and a suitable hyperprior. Again, the connection is made via the Laplace transform, however this time by using $p(s_i)$ directly rather than by reparameterizing with the inverse argument, as done in (120):

$$p(\theta_i) = \int \exp(-\beta\theta_i)\frac{1}{2\beta}p(\beta)d\beta. \tag{125}$$

This means that the Laplace transform of $\frac{1}{s_i}p(s_i)$ yields the effective prior and vice versa. While a Laplacian distribution may not be quite as desirable as a normal distribution (it will typically lead to the solution of a linear program rather than a simple matrix inversion), it is likely to be nonetheless favorable to a direct attempt at computing the mode of the distribution.

Finally, it is worth noting that we can employ the same methods that we used to obtain a normal distribution in $\theta_i$ can be used to transform $p(y_i|t_i)$ into normal distributions combined with a hyperprior:

$$p(y_i|t_i) = \int \frac{\sigma_i}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\sigma_i(y_i - t_i)^2\right) p(\sigma_i)d\sigma_i. \tag{126}$$

The ensuing considerations are completely analogous to the ones in the previous chapter. We will come back to (126) when dealing with general factorizing estimation problems.

## 6.3 Regression with Hyperparameters

As before, we begin with the simple case of regression with additive Gaussian noise, that is $Y|t \sim \mathcal{N}(0, \sigma^2)$. We know that $\boldsymbol{\theta} \sim \mathcal{N}(0, S^{-1})$ where

$S := \operatorname{diag}(s_1, \ldots, s_m)$. Therefore $t = K\boldsymbol{\theta}$ satisfies $t \sim \mathcal{N}(0, K^\top S^{-1} K)$ and finally $Y \sim \mathcal{N}(0, K^\top S^{-1} K + \sigma^2 \mathbf{1})$. Consequently we obtain

$$p(Y|s, \sigma) = (2\pi)^{-\frac{m}{2}} |\tilde{\Sigma}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2} Y^\top \tilde{\Sigma}^{-1} Y\right), \tag{127}$$

where $\tilde{\Sigma} = K^\top S^{-1} K + \sigma^2 \mathbf{1}$. If we wish to determine $p(Y'|Y, s, \sigma)$, we need an estimate of $s'_i$. Assuming that all $s'_i = 0$, the use of the matrix inversion formula plus the application of the Sherman-Morrison-Woodbury formula yields

$$Y'|Y, s, \sigma \sim \mathcal{N}(\mu', \Sigma'). \tag{128}$$

where

$$\mu' = (K')^\top S^{-1} K (\sigma^2 \mathbf{1} + K^\top S^{-1} K)^{-1} Y \text{ and} \tag{129}$$
$$\Sigma' = \sigma^2 \mathbf{1} + (K')^\top (S + \sigma^{-2} K K^\top)^{-1} K'. \tag{130}$$

This follows directly from Example 1. For $K' = K$, i.e., estimation on the training set, the above equations reduce to

$$\mu = K^\top S^{-1} K (\sigma^2 \mathbf{1} + K^\top S^{-1} K)^{-1} Y = \sigma^{-2} K (\sigma^{-2} K^\top K + S)^{-1} K^\top Y. \tag{131}$$

Since elimination of $s, \sigma^2$ by integration is impossible, we resort, as many times before, to the approximation of maximizing the joint density

$$p(Y, s, \sigma) = p(Y|s, \sigma) \prod_{i=1}^m p(s_i) p(\sigma). \tag{132}$$

Assuming a gamma prior not only on $s_i$ but also on $\sigma$, the negative logarithm of (132) can be written as

$$\begin{aligned} -\ln p(Y, s, \sigma) = \frac{1}{2} &\left[\ln \left|\sigma \mathbf{1} + K S^{-1} K^\top\right| + Y^\top \left(\sigma \mathbf{1} + K S^{-1} K^\top\right)^{-1} Y\right] \\ &- \sum_{i=1}^m (a \ln s_i - b s_i) - c \ln \sigma^2 + d\sigma^2 + \text{const}. \end{aligned} \tag{133}$$

Of course, if we set $a = b = c = d = 0$ (flat prior) the terms in (133) vanish and we are left with maximizing the parameters over an improper prior. Note the similarity to logarithmic barrier methods in constrained optimization, for which constrained minimization problems are transformed into unconstrained problems by adding logarithms of the constraints to the initial objective function. In other words, the Gamma distribution can be viewed as a positivity constraint on the hyperparameters $s_i$ and $\sigma^2$.

Differentiating (133) and setting the corresponding terms to 0 leads to the update rules [73]

$$s_i = \frac{1 - s_i \Xi_{ii}}{\mu_i^2}, \tag{134}$$

where $\mu$ is given by (131) and $\Xi := (\sigma^{-2}KK^\top + S)^{-1}$. The quantity $1 - s_i\Xi_{ii}$ is a measure of the degree to which the corresponding parameter $\theta_i$ is "determined" by the data [40]. Likewise we obtain

$$\sigma^2 = \frac{\|\mathbf{y} - K\mu\|^2}{\sum_{i=1}^{m} s_i\Xi_{ii}}. \tag{135}$$

It turns out that many of the parameters $s_i$ tend to infinity during the optimization process. This means that the corresponding distribution of $\theta_i$ is strongly peaked around 0, and we may drop these variables from the optimization process. This speeds up the process as the minimization progresses.

It seems wasteful to first consider the full set of possible functions $k(x_i, x)$, and only then weed out the functions not needed for prediction. We could instead use a greedy method for building up predictors, similar to the greedy strategy employed in Gaussian Processes (Section 4.4). This is the approach in [73], which proposes the following algorithm. After initializing the predictor with a single basis function (the bias, for example), we test whether each new basis function yields an improvement. The latter is achieved by guessing a large initial value $s_i$, and performing one update step. If (134) leads to an increase of $s_i$, we reject the corresponding basis function, otherwise we retain it in the optimization process.

## 6.4 Classification

For classification, we follow a scheme similar to that in Section 3.5. In order to keep matters simple, we only consider the binary classification case. Specifically, we carry out logistic regression by using (14) as a model for the distribution of labels $y_i \in \{\pm 1\}$. As in regression, we use a kernel expansion, this time for the latent variables $t = K\boldsymbol{\theta}$. The negative log density in $Y, \boldsymbol{\theta}$ conditioned on $s, X$ is given by

$$-\ln p(Y, \boldsymbol{\theta}|s, X) = \sum_{i=1}^{m} - \ln p(y_i|t_i(\boldsymbol{\theta})) - \sum_{i=1}^{m} \ln p(\theta_i|s_i) + \text{const. where } t = K\boldsymbol{\theta}. \tag{136}$$

Unlike in regression, however, we cannot minimize (136) explicitly and have to resort to approximate methods, such as the Laplace approximation (see Section 4.1). Computing the first and second derivatives of (136) and using the definitions (70) and (71) yields

$$\partial_{\boldsymbol{\theta}} \left[-\ln p(\boldsymbol{\theta}|\mathbf{y}, \mathbf{s})\right] = Kc + S\boldsymbol{\theta}, \tag{137}$$

$$\partial^2_{\boldsymbol{\theta}} \left[-\ln p(\boldsymbol{\theta}|\mathbf{y}, \mathbf{s})\right] = K^\top CK + S. \tag{138}$$

This allows us to obtain a MAP estimate of $p(\boldsymbol{\theta}|\mathbf{y}, \mathbf{s})$ by iterative application of (69), and we obtain an update rule for $\boldsymbol{\theta}$ in a manner analogous to (72);

$$\boldsymbol{\theta}_{\text{new}} = \boldsymbol{\theta}_{\text{old}} - (K^\top CK + S)^{-1}(Kc + S\boldsymbol{\theta}_{\text{old}}) = (K^\top CK + S)^{-1}K(CK\boldsymbol{\theta}_{\text{old}} - c). \tag{139}$$

If the iteration scheme converges, it will converge to the minimum of the negative log posterior. We next have to provide an iterative method for updating the hyperparameters $\mathbf{s}$ (note that we do not need $\sigma^2$). Since we cannot integrate out $\boldsymbol{\theta}$ explicitly (we had to resort to an iterative method even to obtain the mode of the distribution), it is best to use the Gaussian approximation obtained from (138). This gives an approximation of the value of the posterior distribution $p(\mathbf{s}|\mathbf{y})$ and allows us to apply the update rules developed for regression in classification. Setting $\mu = \boldsymbol{\theta}_{\mathrm{MAP}}$ and $\Xi = (K^\top CK + S)^{-1}$, we can use (134) to optimize $s_i$. See [73] for further detail and motivation.

## 7 Summary

In this paper, we presented an overview over some of the more common techniques of Bayesian estimation, namely Gaussian Processes and the Relevance Vector Machine, and a novel method: Laplacian Processes. Due to the wealth of existing concepts and algorithms developed in Bayesian statistics, it is impossible to give a comprehensive treatment in a single paper. Instead, such a goal would merit writing a book in its own right. We refer the reader to [46, 26, 63, 40, 77, 81] and references therein for further detail.

*Topics We Left Out:* We did not discuss Markov-Chain Monte Carlo methods and their application to Bayesian Estimation [45, 54] as an alternate way of performing Bayesian inference. These work by sampling from the posterior distribution rather than computing an approximation of the mode.

On the model side, the maximum entropy discrimination paradigm [64, 7, 30] is a worthy concept in its own right, powerful enough to spawn a whole family of new inference algorithms both with [30] and without [34] kernels. The main idea is to seek the least informative estimate for prediction purposes. In addition, rather than requiring that a specific function satisfy certain constraints, we require only that the distribution satisfy the constraints *on average.*

Methods such as the Bayes-Point machine [27] and Kernel Billiard [59, 58] can also be used for estimation purposes. The idea behind these methods is to “play billiard” in version space and average over the existing trajectories. The version space is the set of all separating hyperplanes for which the empirical risk vanishes or is bounded by some previously chosen constant. Proponents of this strategy claim rapid convergence due to the good mixing properties of the dynamical system.

Finally, we left the field of graphical models (see for instance [71, 32, 36, 35] and the references therein) completely untouched. These algorithms model the dependency structure between different random variables in a rather explicit fashion and use efficient approximate inference techniques to solve the optimization problems. It is not clear yet, how to combine graphical models with kernels.

*Key Issues:* Topics covered in this paper include deterministic and approximate methods for Bayesian inference, with an emphasis on the Maximum a Posteriori

(MAP) estimate and the treatment of hyperparameters. As a side-effect, one can observe that the minimization of regularized risk is closely related to approximate Bayesian estimation.

One of the first consequences of this link is the connection between Gaussian Processes and Support Vector Machines. While the former are defined in terms of correlations between random variables, the latter are derived from smoothness assumptions regarding the estimate and feature space considerations. This connection also allows one to exchange uniform convergence statements and Bayesian error bounds between both types of reasoning.

As a side effect, this connection also gives rise to a new class of prior, namely those corresponding to $\ell_1$ regularization and linear programming machines. Since the coefficients $\theta_i$ then follow a Laplacian distribution, we name the corresponding stochastic process a *Laplacian Process*. This new point of view allows the derivation of error bars for the estimates in a way that is not easily possible in a statistical learning theory framework. It turns out that this leads to a data dependent prior on the function space.

Finally, the Relevance Vector Machine introduces individual hyperparameters for the distributions of the coefficients $\theta_i$. This makes certain optimization problems tractable (matrix inversion) that otherwise would have remained infeasible (MAP estimate with the Student-t distribution as a prior). We expect that the technique of representing complex distributions by a normal distribution *cum* hyperprior is also a promising approach for other estimation problems.

Taking a more abstract view, we expect a convergence between different estimation algorithms and inference principles derived from risk minimization, Bayesian estimation, and Minimum Description Length concepts. Laplacian Processes and the Relevance Vector Machine are two examples of such convergence. We hope that more such methods will follow in the next few years.

## References

1. K. P. Bennett, A. Demiriz, and J. Shawe-Taylor. A column generation algorithm for boosting. In P. Langley, editor, *Proceedings of the International Conference on Machine Learning*, San Francisco, 2000. Morgan Kaufmann Publishers.
2. K. P. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.
3. C. M. Bishop. *Neural Networks for Pattern Recognition.* Clarendon Press, Oxford, 1995.
4. C. M. Bishop and M. E. Tipping. Variational relevance vector machines. In *Proceedings of 16th Conference on Uncertainty in Artificial Intelligence UAI'2000*, pages 46–53, 2000.
5. P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In J. Shavlik, editor, *Proceedings of the International Conference on Machine Learning*, pages 82–90, San Francisco, California, 1998. Morgan Kaufmann Publishers. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-03.ps.Z.
6. S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *Siam Journal of Scientific Computing*, 20(1):33–61, 1999.

7. T. M. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley and Sons, New York, 1991.
8. H. Cramér. *Mathematical Methods of Statistics.* Princeton University Press, 1946.
9. L. Csató and M. Opper. Sparse representation for Gaussian process models. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 444–450. MIT Press, 2001.
10. A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society B*, 39(1):1–22, 1977.
11. S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195:216–222, 1995.
12. S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representation. Technical report, IBM Watson Research Center, New York, 2000.
13. R. Fletcher. *Practical Methods of Optimization.* John Wiley and Sons, New York, 1989.
14. G. Fung and O. L. Mangasarian. Data selection for support vector machine classifiers. In *Proceedings of KDD'2000*, 2000. also: Data Mining Institute Technical Report 00-02, University of Wisconsin, Madison.
15. A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis.* Chapman and Hall, London, 1995.
16. M. Gibbs and D. J. C. Mackay. Variational Gaussian process classifiers. Technical report, Cavendish Laboratory, Cambridge, UK, 1998.
17. M. N. Gibbs. *Bayesian Gaussian Methods for Regression and Classification.* PhD thesis, University of Cambridge, 1997.
18. Mark Gibbs and David J. C. Mackay. Efficient implementation of Gaussian processes. Technical report, Cavendish Laboratory, Cambridge, UK, 1997. available at http://wol.ra.phy.cam.ac.uk/mng10/GP/.
19. P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization.* Academic Press, 1981.
20. F. Girosi. Models of noise and robust estimates. A.I. Memo 1287, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1991.
21. D. Goldfarb and K. Scheinberg. A product-form cholesky factorization method for handling dense columns in interior point methods for linear programming. Technical report, IBM Watson Research Center, Yorktown Heights, 2001.
22. G. H. Golub and C. F. Van Loan. *Matrix Computations.* John Hopkins University Press, Baltimore, MD, 3rd edition, 1996.
23. I. S. Gradshteyn and I. M. Ryzhik. *Table of integrals, series, and products.* Academic Press, New York, 1981.
24. T. Graepel, R. Herbrich, P. Bollmann-Sdorra, and K. Obermayer. Classification on pairwise proximity data. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 438–444, Cambridge, MA, 1999. MIT Press.
25. D. Haussler. Convolutional kernels on discrete structures. Technical Report UCSC-CRL-99-10, Computer Science Department, UC Santa Cruz, 1999.
26. R. Herbrich. *Learning Kernel Classifiers: Theory and Algorithms.* MIT Press, 2002.
27. Ralf Herbrich, Thore Graepel, and Colin Campbell. Bayes point machines: Estimating the Bayes point in kernel space. In *Proceedings of IJCAI Workshop Support Vector Machines*, pages 23–27, 1999.
28. R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, Cambridge, 1985.

29. P. J. Huber. Robust statistics: a review. *Annals of Statistics*, 43:1041, 1972.
30. T. Jaakkola, M. Meila, and T. Jebara. Maximum entropy discrimination. Technical Report AITR-1668, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1999.
31. T. S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 487–493, Cambridge, MA, 1999. MIT Press.
32. T. S. Jaakkola and M. I. Jordan. Computing upper and lower bounds on likelihoods in untractable networks. In *Proceedings of the 12th Conference on Uncertainty in AI.* Morgan Kaufmann Publishers, 1996.
33. W. James and C. Stein. Estimation with quadratic loss. In *Proceedings of the Fourth Berkeley Symposium on Mathematics, Statistics and Probability*, volume 1, pages 361–380, Berkeley, 1960. University of California Press.
34. T. Jebara and T. Jaakkola. Feature selection and dualities in maximum entropy discrimination. In *Uncertainty In Artificial Intelligence*, 2000.
35. M. I. Jordan and C. M. Bishop. *An Introduction to Probabilistic Graphical Models.* MIT Press, 2002.
36. M. I. Jordan, Z. Gharamani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. In *Learning in Graphical Models*, volume M.I. Jordan, pages 105–162. Kluwer Academic, 1998.
37. M. S. Lewicki and T. J. Sejnowski. Learning nonlinear overcomplete representations for efficient coding. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 556–562, Cambridge, MA, 1998. MIT Press.
38. D. G. Luenberger. *Introduction to Linear and Nonlinear Programming.* Addison-Wesley, Reading, MA, 1973.
39. H. Lütkepohl. *Handbook of Matrices.* John Wiley and Sons, Chichester, 1996.
40. D. J. C. MacKay. *Bayesian Methods for Adaptive Models.* PhD thesis, Computation and Neural Systems, California Institute of Technology, Pasadena, CA, 1991.
41. D. J. C. MacKay. The evidence framework applied to classification networks. *Neural Computation*, 4(5):720–736, 1992.
42. S. Mallat and Z. Zhang. Matching Pursuit in a time-frequency dictionary. *IEEE Transactions on Signal Processing*, 41:3397–3415, 1993.
43. O. L. Mangasarian. Linear and nonlinear separation of patterns by linear programming. *Operations Research*, 13:444–452, 1965.
44. B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal of Computing*, 25(2):227–234, 1995.
45. R. Neal. Priors for infinite networks. Technical Report CRG-TR-94-1, Dept. of Computer Science, University of Toronto, 1994.
46. R. Neal. *Bayesian Learning in Neural Networks.* Springer, 1996.
47. Radford M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical report, Dept. of Computer Science, University of Toronto, 1993. CRG-TR-93-1.
48. B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.
49. M. Opper and O. Winther. Mean field methods for classification with Gaussian processes. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 309–315, Cambridge, MA, 1999. MIT Press.

50. M. Opper and O. Winther. Gaussian processes and SVM: mean field and leave-one-out. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 311–326, Cambridge, MA, 2000. MIT Press.
51. J. Platt. Probabilities for SV machines. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–73, Cambridge, MA, 2000. MIT Press.
52. T. Poggio. On optimal nonlinear associative recall. *Biological Cybernetics*, 19:201–209, 1975.
53. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing (2nd ed.).* Cambridge University Press, Cambridge, 1992. ISBN 0-521-43108-5.
54. C. Rasmussen. *Evaluation of Gaussian Processes and Other Methods for Non-Linear Regression.* PhD thesis, Department of Computer Science, University of Toronto, 1996. ftp://ftp.cs.toronto.edu/pub/carl/thesis.ps.gz.
55. G. Rätsch, S. Mika, and A.J. Smola. Adapting codes und embeddings for polychotomies. In *Neural Information Processing Systems*, volume 15. MIT Press, 2002. to appear.
56. B. D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, Cambridge, 1996.
57. R. T. Rockafellar. *Convex Analysis*, volume 28 of *Princeton Mathematics Series.* Princeton University Press, 1970.
58. P. Ruján and M. Marchand. Computing the Bayes kernel classifier. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 329–347, Cambridge, MA, 2000. MIT Press.
59. Pál Ruján. Playing billiards in version space. *Neural Computation*, 9:99–122, 1997.
60. B. Schölkopf, S. Mika, C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. Smola. Input space vs. feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5):1000–1017, 1999.
61. B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods—Support Vector Learning*, pages 327–352. MIT Press, Cambridge, MA, 1999.
62. B. Schölkopf and A. J. Smola. *Learning with Kernels.* MIT Press, Cambridge, MA, 2002.
63. M. Seeger. Bayesian methods for support vector machines and Gaussian processes. Master's thesis, University of Edinburgh, Division of Informatics, 1999.
64. J. Skilling. *Maximum Entropy and Bayesian Methods.* Cambridge University Press, 1988.
65. A. Smola, B. Schölkopf, and G. Rätsch. Linear programs for automatic accuracy control in regression. In *Ninth International Conference on Artificial Neural Networks*, Conference Publications No. 470, pages 575–580, London, 1999. IEE.
66. A. J. Smola. *Learning with Kernels.* PhD thesis, Technische Universität Berlin, 1998. GMD Research Series No. 25.
67. A. J. Smola and P. L. Bartlett. Sparse greedy Gaussian process regression. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 619–625. MIT Press, 2001.
68. A. J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In P. Langley, editor, *Proceedings of the International Conference on Machine Learning*, pages 911–918, San Francisco, 2000. Morgan Kaufmann Publishers.

69. A.J. Smola and S.V.N. Vishwanathan. Cholesky factorization for rank-$k$ modifications of diagonal matrices. *SIAM Journal of Matrix Analysis*, 2002. submitted.
70. C. Soon-Ong, A.J. Smola, and R.C. Williamson. Superkernels. In *Neural Information Processing Systems*, volume 15. MIT Press, 2002. to appear.
71. D. J. Spiegelhalter and S. L. Lauritzen. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20:579–605, 1990.
72. J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis.* Springer, New York, second edition, 1993.
73. M. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.
74. V. Tresp. A Bayesian committee machine. *Neural Computation*, 12(11):2719–2741, 2000.
75. V. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, 1995.
76. V. Vapnik, S. Golowich, and A. Smola. Support vector method for function approximation, regression estimation, and signal processing. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 281–287, Cambridge, MA, 1997. MIT Press.
77. G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics.* SIAM, Philadelphia, 1990.
78. C. Watkins. Dynamic alignment kernels. CSD-TR-98- 11, Royal Holloway, University of London, Egham, Surrey, UK, 1999.
79. G. N. Watson. *A Treatise on the Theory of Bessel Functions.* Cambridge University Press, Cambridge, UK, 2 edition, 1958.
80. C. K. I. Williams. Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In M. I. Jordan, editor, *Learning and Inference in Graphical Models.* Kluwer Academic, 1998.
81. C. K. I. Williams. Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In Micheal Jordan, editor, *Learning and Inference in Graphical Models*, pages 599–621. MIT Press, 1999.
82. C. K. I. Williams and C. E. Rasmussen. Gaussian processes for regression. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 514–520, Cambridge, MA, 1996. MIT Press.
83. Christoper K. I. Williams and Matthias Seeger. Using the Nystrom method to speed up kernel machines. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 682–688, Cambridge, MA, 2001. MIT Press.
84. Christopher K. I. Williams and David Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI*, 20(12):1342–1351, 1998.
85. T. Zhang. Some sparse approximation bounds for regression problems. In *Proc. 18th International Conf. on Machine Learning*, pages 624–631. Morgan Kaufmann, San Francisco, CA, 2001.