Journal of Machine Learning Research 3 (2002) 233–269 Submitted 3/02; Published 10/02

PAC-Bayesian Generalisation Error Bounds for Gaussian Process Classification

Matthias Seeger SEEGER@DAI.ED.AC.UK
Institute for Adaptive and Neural Computation
University of Edinburgh
5 Forrest Hill, Edinburgh EH1 2QL, UK

Editor: Peter Bartlett

Abstract

Approximate Bayesian Gaussian process (GP) classification techniques are powerful non-parametric learning methods, similar in appearance and performance to support vector machines. Based on simple probabilistic models, they render interpretable results and can be embedded in Bayesian frameworks for model selection, feature selection, etc. In this paper, by applying the PAC-Bayesian theorem of McAllester (1999a), we prove distribution-free generalisation error bounds for a wide range of approximate Bayesian GP classification techniques. We also provide a new and much simplified proof for this powerful theorem, making use of the concept of convex duality which is a backbone of many machine learning techniques. We instantiate and test our bounds for two particular GPC techniques, including a recent sparse method which circumvents the unfavourable scaling of standard GP algorithms. As is shown in experiments on a real-world task, the bounds can be very tight for moderate training sample sizes. To the best of our knowledge, these results provide the tightest known distribution-free error bounds for approximate Bayesian GPC methods, giving a strong learning-theoretical justification for the use of these techniques.

Keywords: Gaussian Processes, Generalisation Error Bounds, PAC-Bayesian Framework, Bayesian Learning, Sparse Approximations, Gibbs Classifier, Kernel Machines, Convex Duality.

1. Introduction

The Bayesian framework for probabilistic inference is widely used all over the statistics and machine learning communities, due to its high flexibility, its ability to render interpretable results and its conceptual simplicity. Within the framework, essential and difficult tasks like model and feature selection have canonical solutions. Complex models for real-world situations can be combined from simple, well-understood components in a structured way. Last, but not least, pitfalls hindering successful generalisation from finite data, such as over-fitting, can be tackled in a clear and principled way, so that Bayesian or approximate Bayesian solutions are typically among the top performers on difficult learning tasks. It is therefore of high theoretical and practical importance to analyse and understand the generalisation capability of (approximate) Bayesian methods. Many analyses so far have concentrated on the case where the true data distribution (stable aspects of which we try to learn) comes from a known family, which is either exactly the model family that the Bayesian method is using, or one which is close in some sense (e.g., Haussler and

Opper, 1997, Haussler et al., 1994, Sollich, 1999). Such analyses are important because they show up the principal limitations of the model family and the induction method, and because they often render close approximations to the true generalisation error we observe on independent test samples. However, they cannot give a guaranteed upper bound on the generalisation error (or other expectations of the true data distribution), because the validity of the whole analysis depends on assumptions that may not hold for the data distribution. PAC analyses[1] of the generalisation capability of a learning technique provide such guaranteed bounds, in the sense that the probability of observing a violation of the bound is shown to be smaller than some *a priori* fixed $\delta > 0$, where the probability is over random draws of the training sample *from the true data distribution*. We can hope to find such non-trivial bounds for finite training sample sizes, because we constrain the sampling process which generates the training set.[2] Recently, a general result was obtained by McAllester (1999a) which allows distribution-free analyses of competitive Bayesian or approximate Bayesian methods: the *PAC-Bayesian theorem*. In this paper, we show how to apply this result to approximate *Bayesian Gaussian process classifiers (GPC)*, in order to obtain data-dependent PAC bounds for these powerful non-parametric methods. The PAC-Bayesian theorem and our results for GPC can be stated in simple and familiar terms and can be proved using elementary concepts only. Furthermore, experiments to be presented here indicate that our bounds can be very tight on real-world classification tasks with moderate training sample sizes, to an extent that they can provide practically meaningful generalisation error guarantees and may even be used for model selection in practice.

The structure of the paper is as follows. In the remainder of this section, we give a brief introduction to Gaussian process models for classification settings, together with fixing our notation. We also state McAllester's PAC-Bayesian theorem, the backbone for our results, for which a simplified proof is given in Appendix A. In the following Section 2, we introduce the class of GP classification techniques we are interested in here and show how to apply the PAC-Bayesian theorem to any method from this class: this is our main result. In Section 3, we instantiate our main result for two particular GPC methods, namely Laplace GPC and sparse greedy GPC. The latter is of especially high practical significance due to its linear scaling with the training set size (our experimental results in Section 4.2 serve as demonstration of its impressive performance). Experimental results on a handwritten digits recognition task are presented in Section 4, testing our main result for the special GPC techniques discussed and comparing it to other state-of-the-art kernel classifier bounds. We close with a discussion in Section 5. The notation we use in this paper is summarised in Appendix B.

1. *PAC* stands for *probably approximately correct*, the framework was introduced by Valiant (1984). In this paper, we use the term *PAC bound* as synonym for "distribution-free large deviation bound": a bound on the probability that an i.i.d. training sample gives rise to a large deviation between empirical and generalisation error. The bound is distribution-free, i.e. holds for any data distribution.
2. Typically, we assume that the training sample is drawn i.i.d. (independently and identically distributed) from the data distribution, but other less restrictive assumptions (e.g., Martingale sequences) are also possible.

1.1 The Binary Classification Problem. PAC Bounds

In the *binary classification problem*, we are given data $S = \{(\boldsymbol{x}_i^S, t_i^S) \mid i = 1, \dots, n\}$, $\boldsymbol{x}_i \in \mathcal{X}$, $t_i \in \{-1, +1\}$, sampled independently and identically distributed (i.i.d.) from an unknown *data distribution* over $\mathcal{X} \times \{-1, +1\}$. Our goal is to compute a *classification function* $\mathcal{X} \to \{-1, +1\}$ from S which has small generalisation error on future test points $\boldsymbol{x}_*$, where $(\boldsymbol{x}_*, t_*)$ is sampled from the data distribution, independently of S. Given an algorithm for computing such functions from samples S, we would like to construct a PAC upper bound on the generalisation error of this method. In the sequel, we denote $X_S = \{\boldsymbol{x}_i^S \mid i = 1, \dots, n\}$, $\boldsymbol{t} = (t_i^S)_i$.

Data-independent (or uniform or *a priori*) PAC bounds ignore aspects of the learning algorithm other than the error of the selected classifier on the training set, and instead constrain the classification function to come from a restricted class of finite complexity in some sense. This restriction is done *a priori*, without looking at the sample S, which allows the difference between empirical error (on S) and generalisation error (this difference is referred to as the *gap* in this paper) to be bounded *uniformly* over all functions in the class, simply because the variability of *all* functions is uniformly restricted. Vapnik-Chervonenkis theory (see Vapnik, 1998) essentially answers the question under which circumstances the gap converges uniformly to 0.

Uniform PAC bounds can answer questions about theoretical learnability of problems, however they are often extremely loose or even trivial in many practically relevant cases. There are two main reasons for this, apart from the distribution-free character of the PAC setting itself. First, the gap bounds do not depend on the observed sample S at all. Whether the particular sample S we encounter matches our prior assumptions or not does not influence the bound value. Second, the gap bound value does not depend on the algorithm used to learn the predictor. It holds uniformly over *all* algorithms which select their classification function from the restricted class, even for a “maximally malicious” algorithm which, knowing the true data distribution, selects a function from the class which *maximises* the generalisation error, subject to a constraint on the empirical error on S. As a consequence, we either choose a very restricted class *a priori* to arrive at a small gap bound for reasonable n, thus typically observing high empirical errors on a nontrivial task, or we live with a gap bound value which is trivial for all practically interesting sample sizes n. Both options are not tolerable from a practical viewpoint.

Data-dependent (or *a posteriori*) PAC bounds on the difference between empirical and generalisation error depend on the sample S. The idea is that prior knowledge about the unknown data distribution is used to introduce a weighting in the bounding technique which is biased towards our expectations. Namely, if it turns out that the data distribution matches our expectations rather closely, as judged by an empirical divergence measure which can be evaluated on the sample S, the gap bound value will be small (the *lucky* case).[3] If we are grossly wrong, the bound can be large, usually trivial. At this point, the notion of two *different* sets of assumptions we are working with becomes most clear:

3. The concept of “luckiness” has been introduced to Statistical Learning Theory by Shawe-Taylor et al. (1998). We think that it is very much related to the concept of using prior distributions which penalise complex models (so-called “Occam’s Razor” priors); another good reason for applying PAC analysis to (approximate) Bayesian methods, in which luckiness can be formulated by the well-studied devices of modelling and prior assessment.

- In order to construct a classification method and a data-dependent bound for it, we follow *Bayesian modelling assumptions*: Available prior knowledge is encoded, within feasibility constraints, into a probabilistic model and prior distributions. The extent to which the unknown data distribution is compatible with these assumptions will in general determine the accuracy of the method and the observed tightness (yet not the validity) of the bound.

- The statement of the bound holds under standard *PAC assumptions*: We are given an i.i.d. training sample from the data distribution which is otherwise completely unknown.

Apart from the data dependency, our bound also concentrates *only* on the algorithm we are interested in. Even if this algorithm selects a classification function from some class or uses a mixture of such functions, the bound is specific to the way in which this is done. While a uniform bound merely suggests to select, from within the restricted class, a classifier which minimises the empirical error, in data-dependent bounds we have a trade-off between empirical error and the quality of the match between our prior assumptions and the classification function we construct. We know of no interesting real-world learning problem which comes without any sort of prior knowledge, and most of these problems are at least partly "non-malicious" in the sense that using this prior knowledge improves performance instead of deteriorating it. From this perspective, data-dependent bounds for specific algorithms are most promising for providing practically meaningful generalisation error guarantees.

1.2 The Approach. Gaussian Process Models

Recall from the previous subsection that in order to come up with a meaningful data-dependent PAC bound, we have to formalise our prior knowledge about the task in some concrete way, so that we can construct data-dependent complexity measures for the set of classification functions. The simplest way to do this is to *model* the relationship $\boldsymbol{x} \to t$.

A binary classification model can be seen as a probabilistic formulation of the relations between the variables $\boldsymbol{x} \to y \to t$, where the input variable $\boldsymbol{x} \in \mathcal{X}$, the latent output $y \in \mathbb{R}$ and the observable target $t \in \{-1, +1\}$. Learning and generalisation works by assuming that the latent relationship $\boldsymbol{x} \to y$ is smooth and regular in some sense, however these regularities are obscured by noise; our task is then to separate the structure from the noise.[4] Note that in this paper, we are interested only in *discrimination models*, i.e. we do not attempt to model the data distribution over input points $\boldsymbol{x}$. Distributions such as the data likelihood will always be conditioned on the corresponding input datapoints, although for simplicity this is not made explicit in the notation. Discrimination models typically are more robust and outperform complete models for the whole joint data distribution on tasks where the true input distribution cannot be identified well given the data. In general, a (discrimination) model is decomposed into some kind of family for the latent function $\boldsymbol{x} \mapsto y(\boldsymbol{x})$ and a classification noise model $P(t \mid y)$. From the Bayesian viewpoint, $y(\cdot)$ is a

4. In this context, it is largely irrelevant whether there really *exists* a smooth, underlying latent function. Being restricted to finite data, we cannot test such a hypothesis anyway. The perspective is positivistic: the model is "true" if we can use it to predict the future well.

random function, i.e., our knowledge about it will always remain uncertain to some extent. A common noise model is based on the Bernoulli distribution, with $P(t \,|\, y) = \sigma(ty)$, where $\sigma(u) = (1 + \exp(-u))^{-1}$ is the logistic function. Here, the latent function $y(\cdot)$ models the logit $\log(P(t = +1 \,|\, \boldsymbol{x})/P(t = -1 \,|\, \boldsymbol{x}))$, which is why $P(t \,|\, y)$ is often referred to as *logit noise*. Note that if, for a test point $\boldsymbol{x}_*$, we knew the true logit $y_{\text{true}}(\boldsymbol{x}_*)$, then the most probable target t_* at $\boldsymbol{x}_*$ is $\operatorname{sgn} y_{\text{true}}(\boldsymbol{x}_*)$. Thus, a natural way to do classification within this model-based framework is to estimate the latent function $y(\cdot)$ and then use the classifier $\boldsymbol{x} \mapsto \operatorname{sgn} y(\boldsymbol{x})$. For more information about such discrimination models (see McCullach and Nelder, 1983, Green and Silverman, 1994).

The *parametric* modelling approach imposes a family of candidates $\{\boldsymbol{x} \mapsto y(\boldsymbol{x} \,|\, \boldsymbol{w})\}$ for $y(\cdot)$, where $\boldsymbol{w}$ is a parameter vector, determining the function $y(\cdot \,|\, \boldsymbol{w})$. Examples are linear models (i.e. $y(\boldsymbol{x} \,|\, \boldsymbol{w}) = \boldsymbol{v}^T\boldsymbol{x} + w_0,\ \boldsymbol{w} = (\boldsymbol{v}^T\, w_0)^T$) or multi-layer perceptrons. If we place a *prior distribution* $P(\boldsymbol{w})$ on the parameter vector, the model is completely specified and encodes our prior assumptions about the unknown data distribution. The *non-parametric* modelling approach differs from this, by placing a distribution directly on $y(\cdot)$, without assuming its membership in a fixed parametric family. Such a random process distribution is often constructed implicitly, by defining distributions over $y(X)$ for every finite subset $X \subset \mathcal{X}$. Here, we use the *Matlab* notation, i.e. if $X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_q\}$, then $y(X) = (y(\boldsymbol{x}_1) \dots y(\boldsymbol{x}_q))^T$. A *Gaussian process (GP)* is a random process $y(\cdot)$ over $\mathcal{X}$ whose finite-dimensional marginal distributions are normal, i.e. for every finite set $X \subset \mathcal{X}$ the corresponding random vector $y(X)$ is Gaussian. Furthermore, marginalisation has to be consistent, in the sense that for two overlapping finite sets X_1, X_2, the marginal distributions of $y(X_1)$ and $y(X_2)$ on $X_1 \cap X_2$ have to be the same. The process is essentially determined by a mean function $m(\boldsymbol{x}) = \mathrm{E}[y(\boldsymbol{x})]$ and a covariance kernel $K(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \mathrm{E}[(y(\boldsymbol{x}) - m(\boldsymbol{x}))(y(\tilde{\boldsymbol{x}}) - m(\tilde{\boldsymbol{x}}))]$. In the special case $m(\boldsymbol{x}) \equiv 0$, we refer to $y(\boldsymbol{x})$ as a *zero-mean Gaussian process*. Note that in this case, $K(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \mathrm{E}[y(\boldsymbol{x})y(\tilde{\boldsymbol{x}})]$. By placing a zero-mean Gaussian process prior on the latent function $y(\cdot)$, we can specify a non-parametric model in which the choice of the kernel K encodes our prior assumptions about the unknown data distribution.[5] Note that this definition of a GP is nothing more than the mathematically rigorous generalisation of a multivariate Gaussian random vector if $\mathcal{X}$ is infinite: vectors become functions $\mathcal{X} \to \mathbb{R}$, random vectors become random processes, and positive definite matrices become operators with positive definite kernels. In fact, for classification purposes we always condition on a finite number of input points from $\mathcal{X}$ and have to deal with finite-dimensional Gaussians only, yet the kernel is what associates observations with each other, allows us to generalise to future test points, draw decision boundaries, etc. For an introduction to Gaussian process models in the Bayesian context see (Williams, 1997). We will show in Section 2.1 how approximate Bayesian predictions for GP classification models can be obtained.

Let us finally introduce some notation. For two finite sets X_1, X_2 of input points, let $\boldsymbol{K}(X_1, X_2)$ be the kernel matrix, i.e. $\boldsymbol{K}(X_1, X_2) = (K(\boldsymbol{x}_i^{(1)}, \boldsymbol{x}_j^{(2)}))_{i,j}$, where $\boldsymbol{x}_i^{(k)}$ runs over

5. This prior is reasonable if we know that the classes have equal prior probability, as we will assume in this paper for simplicity. The general case can be treated by a straightforward parametric extension of the model, for which our results remain valid.

the points in X_k. Define $\boldsymbol{K}(X) = \boldsymbol{K}(X, X)$. The kernel matrix over the training inputs is denoted $\boldsymbol{K}_S = \boldsymbol{K}(X_S)$. In the sequel, we will always assume that $\boldsymbol{K}_S$ is positive definite.[6]

1.3 The PAC-Bayesian Theorem

In this subsection, we introduce the PAC-Bayesian theorem of McAllester (1999a), where in this paper we restrict ourselves to the binary classification scenario.

Suppose we are given a function class $\{y(\cdot \,|\, \boldsymbol{w})\}$ parameterised by $\boldsymbol{w}$. Many readers will be most familiar with this parametric formulation, however, note that $\boldsymbol{w}$ need not be a finite-dimensional vector. For example, in our application to non-parametric models below we will identify $\boldsymbol{w}$ with the function $y(\cdot \,|\, \boldsymbol{w})$ itself. In the binary classification setting defined in Section 1.1, it is understood that $y(\boldsymbol{x} \,|\, \boldsymbol{w})$ predicts $t(\boldsymbol{x}) = \operatorname{sgn} y(\boldsymbol{x} \,|\, \boldsymbol{w})$. A certain type of classifier, called *Gibbs classifier*, depends on the hypothesis class as well as on a distribution $Q(\boldsymbol{w})$ over parameter vectors. Namely, given a test point $\boldsymbol{x}_*$, the Gibbs classifier predicts the corresponding target by first sampling $\boldsymbol{w} \sim Q(\boldsymbol{w})$, then returning $t_* = \operatorname{sgn} y(\boldsymbol{x}_*|\boldsymbol{w})$, plugging in the parameter vector just sampled. Note that a Gibbs classifier has a probabilistic element, i.e. requires coin tosses for prediction. Note also that if the targets of several test points are to be predicted, the parameter vectors sampled for this purpose are independent[7]. This is in contrast to the more familiar *Bayes classifier* which (in the case of our classification model) predicts $t_* = \operatorname{sgn} \mathrm{E}_{\boldsymbol{w} \sim Q}[y(\boldsymbol{x}_* \,|\, \boldsymbol{w})]$. Another type of rule, called *Bayes voting classifier* here, predicts $t_* = \operatorname{sgn} \mathrm{E}_{\boldsymbol{w} \sim Q}[\operatorname{sgn} y(\boldsymbol{x}_* \,|\, \boldsymbol{w})]$, i.e. "votes" over classifiers $\operatorname{sgn} y(\boldsymbol{x}_* \,|\, \boldsymbol{w})$ instead of averaging discriminants $y(\boldsymbol{x}_* \,|\, \boldsymbol{w})$.

McAllester's PAC-Bayesian theorem deals with Gibbs classifiers for which the distribution $Q(\boldsymbol{w})$ may depend on the training sample S, which is why $Q(\boldsymbol{w})$ is sometimes referred to as a "posterior distribution". In order to eliminate the probabilistic element in the Gibbs classifier itself, the bound is on the gap between expected generalisation error and expected empirical error, where the expectation is over $Q(\boldsymbol{w})$. The theorem can be configured by a prior distribution $P(\boldsymbol{w})$ over parameter vectors, and the gap bound term depends most strongly on the *relative entropy*

$$\mathrm{D}[Q \,\|\, P] = \mathrm{E}_{\boldsymbol{w} \sim Q(\boldsymbol{w})} \left[\log \frac{dQ(\boldsymbol{w})}{dP(\boldsymbol{w})} \right] \tag{1}$$

between $Q(\boldsymbol{w})$ and the prior $P(\boldsymbol{w})$. The relative entropy as a measure of deviation between two distributions is well-founded in information theory, statistics and many other fields (see Cover and Thomas, 1991). It arises naturally in maximum likelihood estimation and variational Bayesian approximations and is certainly one of the most important concepts in machine learning. Here, we assume that $Q(\boldsymbol{w})$ and $P(\boldsymbol{w})$ are absolutely continuous w.r.t.

6. It is possible to work with covariance kernels which have a non-zero null space and which can give rise to singular kernel matrices (for example spline kernels, see Wahba, 1990), however this requires the use of semi-parametric models which is not pursued here. Note that $\boldsymbol{K}_S$ will typically have a *numerical rank* smaller than n, i.e. will often be ill-conditioned, and one has to be careful to employ appropriate numerically stable techniques.
7. Readers familiar with *Markov chain Monte Carlo* methods will note the similarity with a MCMC approximation (based on one sample of $\boldsymbol{w}$ only) of the corresponding Bayes classifier for $Q(\boldsymbol{w})$. The difference is that typically in MCMC, the sample representing the posterior $Q(\boldsymbol{w})$ is retained and used for many predictions, while in the Gibbs classifier, we use each posterior sample only once.

some positive measure, and $dQ(\boldsymbol{w})/dP(\boldsymbol{w})$ is the Radon-Nikodym derivative (e.g., Ihara, 1993, Sect. 1.4) of $Q(\boldsymbol{w})$ w.r.t. $P(\boldsymbol{w})$. If $Q(\boldsymbol{w})$ is not absolutely continuous w.r.t. $P(\boldsymbol{w})$, i.e. if there is a null set of $P(\boldsymbol{w})$ which is not a null set of $Q(\boldsymbol{w})$, we define $\mathrm{D}[Q \,\|\, P] = \infty$. Let us give some examples. If the mass of $\boldsymbol{w}$ is concentrated on a finite set of size L, say $\{1, \dots, L\}$, the dominating measure is the counting measure, $Q(\boldsymbol{w})$ and $P(\boldsymbol{w})$ are finite distributions and

$$\mathrm{D}[Q \,\|\, P] = \sum_{l=1}^{L} \mathrm{Pr}_Q\{\boldsymbol{w} = l\} \log \frac{\mathrm{Pr}_Q\{\boldsymbol{w} = l\}}{\mathrm{Pr}_P\{\boldsymbol{w} = l\}}. \tag{2}$$

If $\boldsymbol{w} \in \mathbb{R}^m$, the dominating measure is typically the Lebesgue measure $d\boldsymbol{w}$, and

$$\mathrm{D}[Q \,\|\, P] = \mathrm{E}_{\boldsymbol{w} \sim Q(\boldsymbol{w})} \left[\log \frac{Q(\boldsymbol{w})}{P(\boldsymbol{w})} \right].$$

Recall that for simplicity we use the same notation for a distribution and its density w.r.t. $d\boldsymbol{w}$, i.e. $Q(\boldsymbol{w}) = dQ/d\boldsymbol{w}$. For the formulation of Theorem 1, we require (as a special case of Equation 2) the relative entropy between two Bernoulli variables (skew coins) with probabilities of heads $q,\ p$,

$$\mathrm{D}_{\mathrm{Ber}}[q \,\|\, p] = q \log \frac{q}{p} + (1 - q) \log \frac{1 - q}{1 - p}. \tag{3}$$

$\mathrm{D}_{\mathrm{Ber}}$ is convex in (q, p), furthermore $p \mapsto \mathrm{D}_{\mathrm{Ber}}[q \,\|\, p]$ is strictly monotonically increasing for $p \geq q$, mapping $[q, 1)$ to $[0, \infty)$. Thus, the following function

$$\mathrm{D}_{\mathrm{Ber}}^{-1}(q, \varepsilon) = t \ \text{ s.t. } \ \mathrm{D}_{\mathrm{Ber}}[q \,\|\, q + t] = \varepsilon, \ t \geq 0 \tag{4}$$

is well-defined for $q \in [0, 1)$ and $\varepsilon \geq 0$. We also define $\mathrm{D}_{\mathrm{Ber}}^{-1}(q, \infty) = 1 - q$. Note that, due to the convexity of $\mathrm{D}_{\mathrm{Ber}}$, we can compute $\mathrm{D}_{\mathrm{Ber}}^{-1}(q, \varepsilon)$ easily using Newton's algorithm. It is clear by definition that for $\varepsilon \geq 0,\ t \in [0, 1 - q)$:

$$t \geq \mathrm{D}_{\mathrm{Ber}}^{-1}(q, \varepsilon) \iff \mathrm{D}_{\mathrm{Ber}}[q \,\|\, q + t] \geq \varepsilon. \tag{5}$$

Suppose we are given an arbitrary prior distribution $P(\boldsymbol{w})$ over parameter vectors, and we choose a confidence parameter $\delta \in (0, 1)$. Then, the following result holds.

Theorem 1 (PAC-Bayesian theorem (McAllester, 1999a)) *For any data distribution over $\mathcal{X} \times \{-1, +1\}$, we have that the following bound holds, where the probability is over random i.i.d. samples $S = \{(\boldsymbol{x}_i^S, t_i^S) \mid i = 1, \dots, n\}$ of size n drawn from the data distribution:*

$$\mathrm{Pr}_S \left\{ \mathrm{gen}(Q) > \mathrm{emp}(S, Q) + \mathrm{D}_{\mathrm{Ber}}^{-1}(\mathrm{emp}(S, Q), \varepsilon(\delta, n, P, Q)) \ \text{ for some } Q \right\} \leq \delta. \tag{6}$$

Here, $Q = Q(\boldsymbol{w})$ is an arbitrary "posterior" distribution over parameter vectors, which may depend on the sample S and on the prior P. Furthermore,

$$\begin{aligned}
\mathrm{emp}(S, Q) &= \mathrm{E}_{\boldsymbol{w} \sim Q(\boldsymbol{w})} \left[\frac{1}{n} \sum_{i=1}^{n} \mathrm{I}_{\{\mathrm{sgn}\, y(\boldsymbol{x}_i^S \,|\, \boldsymbol{w}) \neq t_i^S\}} \right], \\
\mathrm{gen}(Q) &= \mathrm{E}_{\boldsymbol{w} \sim Q(\boldsymbol{w})} \left[\mathrm{E}_{(\boldsymbol{x}_*, t_*)} \left[\mathrm{I}_{\{\mathrm{sgn}\, y(\boldsymbol{x}_* \,|\, \boldsymbol{w}) \neq t_*\}} \right] \right], \\
\varepsilon(\delta, n, P, Q) &= \frac{1}{n} \left(\mathrm{D}[Q \,\|\, P] + \log \frac{n + 1}{\delta} \right).
\end{aligned}$$

Here, $\mathrm{emp}(S,Q)$ *is the expected empirical error,* $\mathrm{gen}(Q)$ *the expected generalisation error of the Gibbs classifier based on* $Q(\boldsymbol{w})$ *(note that the probability in* $\mathrm{gen}(Q)$ *is over* $(\boldsymbol{x}_*, t_*)$ *drawn from the data distribution, independently of the sample* S*).* $\mathrm{D}_{\mathrm{Ber}}^{-1}(q,\varepsilon)$ *is defined by (4), and* $\mathrm{D}[Q \,\|\, P]$ *denotes the relative entropy between the distributions* Q *and* P*, as defined in (1).*

Note that McAllester's theorem applies more generally to bounded loss functions and makes use of Hoeffding's inequality for bounded variables. However, for the special case of zero-one loss, we can use techniques tailored for binomial variables which give considerably tighter results than Hoeffding's bound if the expected empirical error of the Gibbs classifier is small. This version of McAllester's theorem is proved in (Langford and Seeger, 2001). The proof of Theorem 1 we present here in Appendix A, is considerably simpler and more direct than the arguments used by McAllester (1999a) and McAllester (2002). It is based on the concept of *convex duality* which is itself a cornerstone of many techniques used in machine learning, such as the expectation maximisation algorithm, certain variational approximations to Bayesian inference and primal-dual optimisation schemes.

We should also stress once more, in accordance with what has been said in Section 1.1, that the PAC-Bayesian theorem does *not* require the true data distribution to be constrained in any way depending on the prior $P(\boldsymbol{w})$ and the model class. Other non-PAC analyses would probably require that the conditional data distribution is equal to $\prod_i P(t_i^S \,|\, y(\boldsymbol{x}_i^S, \boldsymbol{w}_*))$, $\boldsymbol{w}_* \sim P(\boldsymbol{w}_*)$, and under this restriction it might be possible to prove stronger results than the PAC-Bayesian theorem (given that the model class is not too large).

1.3.1 Extension to the Bayes Classifier

In most situations in practice, when comparing Gibbs and Bayes classifier for the same posterior distribution $Q(\boldsymbol{w})$ directly, it turns out that the Bayes variant can be computed more efficiently and often performs better than the Gibbs variant. Therefore, it would be of high interest to obtain a PAC-Bayesian theorem for Bayes classifiers as well. For fixed $(\boldsymbol{x}_*, t_*)$, define the errors of Gibbs, Bayes and Bayes voting classifiers as

$$
\begin{aligned}
e_{\mathrm{Gibbs}}(\boldsymbol{x}_*, t_*) &= \mathrm{E}_{\boldsymbol{w}\sim Q}[\mathrm{I}_{\{\mathrm{sgn}\, y(\boldsymbol{x}_* \,|\, \boldsymbol{w}) \neq t_*\}}], \\
e_{\mathrm{Bayes}}(\boldsymbol{x}_*, t_*) &= \mathrm{I}_{\{\mathrm{sgn}\, \mathrm{E}_{\boldsymbol{w}\sim Q}[y(\boldsymbol{x}_* \,|\, \boldsymbol{w})] \neq t_*\}}, \\
e_{\mathrm{Vote}}(\boldsymbol{x}_*, t_*) &= \mathrm{I}_{\{\mathrm{sgn}\, \mathrm{E}_{\boldsymbol{w}\sim Q}[\mathrm{sgn}\, y(\boldsymbol{x}_* \,|\, \boldsymbol{w})] \neq t_*\}}.
\end{aligned}
$$

Furthermore, for $A \in \{\mathrm{Gibbs}, \mathrm{Bayes}, \mathrm{Vote}\}$, define $e_A = \mathrm{E}_{(\boldsymbol{x}_*, t_*)}[e_A(\boldsymbol{x}_*, t_*)]$ where the expectation is over the data distribution. It is easy to relate e_{Vote} and e_{Gibbs}, by noting that if $e_{\mathrm{Vote}}(\boldsymbol{x}_*, t_*) = 1$, then $e_{\mathrm{Gibbs}}(\boldsymbol{x}_*, t_*) \geq 1/2$, thus $e_{\mathrm{Vote}} \leq 2\, e_{\mathrm{Gibbs}}$ (as remarked in Herbrich, 2001, lemma 5.3). The Bayes and the Bayes voting classifier are different rules in general, but in the special case that for each fixed $(\boldsymbol{x}_*, t_*)$, the distribution of $y(\boldsymbol{x}_* \,|\, \boldsymbol{w})$, $\boldsymbol{w} \sim Q$ is symmetric around its mean, one can easily show that they are identical (thanks to Manfred Opper for pointing this out). Namely, fix $\boldsymbol{x}_*$, write $y_* = y(\boldsymbol{x}_* \,|\, \boldsymbol{w})$, $\langle y_* \rangle = \mathrm{E}_{\boldsymbol{w}\sim Q}[y(\boldsymbol{x}_* \,|\, \boldsymbol{w})]$ and let $u = y_* - \langle y_* \rangle$. The latter has a distribution which is symmetric with mean 0. Now, the product of the predictions of t_* by the Bayes and the Bayes voting variant is

$$
\mathrm{sgn}\, \mathrm{E}\left[\mathrm{sgn}\left(\langle y_* \rangle y_*\right)\right] = \mathrm{sgn}\, \mathrm{E}\left[\mathrm{sgn}\left(\langle y_* \rangle^2 + \langle y_* \rangle u\right)\right],
$$

which is 1 if $\langle y_* \rangle \neq 0$. Since for $\langle y_* \rangle = 0$, both variants predict $\operatorname{sgn} 0$, we see that they always predict the same t_*.

Combining these observations, we see that under the symmetry condition we have that $e_{\text{Bayes}} \leq 2\, e_{\text{Gibbs}}$, thus in this case Theorem 1 applies to the Bayes classifier as well. However, this is not really the result we are ideally looking for. Namely, given special distributional families for P and Q, we would like to obtain an analogue of Theorem 1 which results in a bound for e_{Bayes} which is the same or better than what we know for e_{Gibbs}, or at least no more than $1+\varepsilon$ times the e_{Gibbs} bound, where $\varepsilon \ll 1$. Very recently, Meir and Zhang (2002) obtained a strong PAC-Bayesian margin bound for the Bayes voting classifier, combining a new inequality based on Rademacher complexities with convex duality (as in our proof).

2. The PAC-Bayesian Theorem for Approximate Bayesian Gaussian Process Classification

In this section, we derive our main result: the application of the PAC-Bayesian Theorem 1 to a wide class of approximate Bayesian Gaussian process classification methods. We begin in Section 2.1 by introducing the Bayesian inference problem for binary GP classification and the class of approximate solutions we are interested in here. Methods in this class approximate the true intractable predictive posterior process by a Gaussian one. In Section 2.2, we show how to compute the relative entropy (1) between such prior and posterior Gaussian processes. Finally, we state and discuss our main result (Theorem 2) in Section 2.3.

2.1 Approximate Bayesian Gaussian Process Classification

In this section, we introduce the Bayesian inference problem over GP classification models and the class of approximate methods which we are concerned with in this paper. This class is very broad and encompasses almost all Bayesian GPC approximations we know of.

Given some data S with input points $X_S = \{\boldsymbol{x}_i^S \,|\, i = 1, \dots, n\}$ and targets $\boldsymbol{t} = (t_i^S)_i$, let $\boldsymbol{y}_S = y(X_S) \in \mathbb{R}^n$. The Bayesian posterior distribution for $\boldsymbol{y}_S$ is given by $P(\boldsymbol{y}_S \,|\, S) \propto P(S \,|\, \boldsymbol{y}_S) P(\boldsymbol{y}_S)$, where $P(\boldsymbol{y}_S) = N(\boldsymbol{0}, \boldsymbol{K}_S)$, $\boldsymbol{K}_S = \boldsymbol{K}(X_S)$ by the GP prior, and

$$P(S \,|\, \boldsymbol{y}_S) = \prod_{i=1}^{n} P(t_i^S \,|\, y_i^S), \quad \boldsymbol{y}_S = (y_i^S)_i$$

is the (conditional) likelihood. Unfortunately, due to the non-Gaussian noise distribution $P(t \,|\, y)$, it is in general intractable to work with the exact posterior $P(\boldsymbol{y}_S \,|\, S)$ in order to do predictions. The class of approximations we are interested in here replaces $P(\boldsymbol{y}_S \,|\, S)$ by a Gaussian distribution $Q(\boldsymbol{y}_S \,|\, S)$. Once we have done this replacement, no further approximations are necessary, because the corresponding approximation of the predictive or posterior process turns out to be Gaussian as well. Namely, for any finite (ordered) set $X \subset \mathcal{X}$, let $\boldsymbol{y} = y(X)$. Now, by $\boldsymbol{y} \setminus \boldsymbol{y}_S$, we denote the (ordered) collection of variables obtained by deleting all components in $\boldsymbol{y}$ which correspond to input points of X that occur in X_S. $\boldsymbol{y}_S \setminus \boldsymbol{y}$ is defined analogously. Now, define the distribution of $\boldsymbol{y}$ to be

$$Q(\boldsymbol{y} \,|\, S) = \int P(\boldsymbol{y} \setminus \boldsymbol{y}_S \,|\, \boldsymbol{y}_S) Q(\boldsymbol{y}_S \,|\, S)\, d(\boldsymbol{y}_S \setminus \boldsymbol{y}). \tag{7}$$

Here, we follow the usual convention that densities over an empty set of variables are taken to be constant 1, and integrals over an empty set of variables are simply not done. This definition is a consequence of our data model, if we plug in $Q(\boldsymbol{y}_S \,|\, S)$ for $P(\boldsymbol{y}_S \,|\, S)$. Namely, first $Q(\boldsymbol{y}, \boldsymbol{y}_S \,|\, S) = P(\boldsymbol{y} \setminus \boldsymbol{y}_S \,|\, \boldsymbol{y}_S) Q(\boldsymbol{y}_S \,|\, S)$, from which we obtain (7) by integrating over $\boldsymbol{y}_S \setminus \boldsymbol{y}$. It is clear that $Q(\boldsymbol{y} \,|\, S)$ is Gaussian, and the consistency requirement can be checked straightforwardly, thus we have defined a Gaussian process approximating the true intractable posterior process, which can be used for (approximate) prediction as follows.

Suppose that

$$Q(\boldsymbol{y}_S \,|\, S) = N(\boldsymbol{y}_S \,|\, \boldsymbol{K}_S \hat{\boldsymbol{\alpha}}_S, \boldsymbol{\Sigma}_S) \tag{8}$$

is the posterior approximation. Here, the parameters $\hat{\boldsymbol{\alpha}}_S \in \mathbb{R}^n$ and $\boldsymbol{\Sigma}_S \in \mathbb{R}^{n,n}$ can be chosen in an arbitrary way, given that $\boldsymbol{\Sigma}_S$ and $\boldsymbol{K}_S$ are positive definite. In order to predict $y_* = y(\boldsymbol{x}_*)$ at a test point $\boldsymbol{x}_*$, we need to compute $Q(y_* \,|\, \boldsymbol{x}_*, S) = N(y_* \,|\, \mu(\boldsymbol{x}_*), \sigma^2(\boldsymbol{x}_*))$. Let $\boldsymbol{k}(\boldsymbol{x}_*) = \boldsymbol{K}(X_S, \{\boldsymbol{x}_*\})$. By computing the joint Gaussian $P(y_*, \boldsymbol{y}_S)$ and conditioning on $\boldsymbol{y}_S$, it is easy to see that

$$P(y_* \,|\, \boldsymbol{y}_S) = N(y_* \,|\, \boldsymbol{k}(\boldsymbol{x}_*)^T \boldsymbol{K}_S^{-1} \boldsymbol{y}_S,\ K(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{k}(\boldsymbol{x}_*)^T \boldsymbol{K}_S^{-1} \boldsymbol{k}(\boldsymbol{x}_*)).$$

From this equation and (7) we see that $y_* \sim Q(y_* \,|\, \boldsymbol{x}_*, S)$ has the same distribution as $r + \boldsymbol{k}(\boldsymbol{x}_*)^T \boldsymbol{K}_S^{-1} \boldsymbol{y}_S$, where $r \sim N(r \,|\, 0, K(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{k}(\boldsymbol{x}_*)^T \boldsymbol{K}_S^{-1} \boldsymbol{k}(\boldsymbol{x}_*))$ independent of $\boldsymbol{y}_S \sim Q(\boldsymbol{y}_S \,|\, S)$. Thus, by standard theorems of Normal theory (e.g., Mardia et al., 1979, Chap. 3), we arrive at

$$\begin{aligned} \mu(\boldsymbol{x}_*) &= \boldsymbol{k}(\boldsymbol{x}_*)^T \hat{\boldsymbol{\alpha}}_S, \quad \sigma^2(\boldsymbol{x}_*) = K(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{k}(\boldsymbol{x}_*)^T \boldsymbol{M}_S \boldsymbol{k}(\boldsymbol{x}_*), \\ \boldsymbol{M}_S &= \boldsymbol{K}_S^{-1} - \boldsymbol{K}_S^{-1} \boldsymbol{\Sigma}_S \boldsymbol{K}_S^{-1}. \end{aligned} \tag{9}$$

The Equations (9) can typically be somewhat simplified for concrete $\hat{\boldsymbol{\alpha}}_S$, $\boldsymbol{\Sigma}_S$ (as chosen by one of the approximate GPC methods we discuss below). Note that a predictive distribution for the target t_*, i.e. $Q(t_* \,|\, \boldsymbol{x}_*, S)$, can be obtained by averaging the noise distribution $P(t_* \,|\, y_*)$ over $Q(y_* \,|\, \boldsymbol{x}_*, S)$. This is a simple one-dimensional integral which can be approximated using a numerical quadrature rule. The corresponding approximate Bayes classifier, i.e. the rule which chooses t_* to maximise $Q(t_* \,|\, \boldsymbol{x}_*, S)$ is given by $\boldsymbol{x}_* \mapsto \operatorname{sgn} \mu(\boldsymbol{x}_*)$, due to the symmetry of $Q(y_* \,|\, \boldsymbol{x}_*, S)$ around its mean and the fact that $P(t_* = +1 \,|\, y_*) - 1/2$ is an odd function. This rule depends on the predictive mean $\mu(\boldsymbol{x}_*)$ only, while it will turn out below that the evaluation of the corresponding approximate Gibbs classifier requires the evaluation of $\sigma^2(\boldsymbol{x}_*)$ as well. Thus, in the context of the GPC approximations we are interested in here, the Bayes classifier can usually be evaluated more efficiently than the Gibbs variant if extra information such as $Q(t_* \,|\, \boldsymbol{x}_*, S)$ is not required.

Approximation methods in the class we consider here differ in their choice of the parameters of $Q(\boldsymbol{y}_S \,|\, S)$. Optimally, these parameters are chosen such that the true predictive process $P(t_* \,|\, \boldsymbol{x}_*, S)$ is closest to $Q(t_* \,|\, \boldsymbol{x}_*, S)$ in relative entropy. A more feasible choice is, however, to match the predictive processes $P(y_* \,|\, \boldsymbol{x}_*, S)$ and $Q(y_* \,|\, \boldsymbol{x}_*, S)$ in this way, which is equivalent to the maximum likelihood projection of $P(y_* \,|\, \boldsymbol{x}_*, S)$ onto the family of Gaussian processes of the form shown in (7). The optimal choice of parameters requires matching of moments between $P(\boldsymbol{y}_S \,|\, S)$ and $Q(\boldsymbol{y}_S \,|\, S)$, and we can see from (9) that for this choice, the (approximate) Bayes classifier depends on the posterior mean of $P(\boldsymbol{y}_S \,|\, S)$

only. However, this mean is difficult to find (it can be approximated using MCMC sampling techniques, see Neal, 1997), and most approximations settle for other parameters of $Q(\boldsymbol{y}_S \,|\, S)$.

In the context of this paper, we are interested in the posterior Gibbs rather than the posterior Bayes classifier. The former predicts the target t_* at a test point $\boldsymbol{x}_*$ by sampling $y_* \sim Q(y_* \,|\, \boldsymbol{x}_*, S)$ from the approximate predictive distribution, then outputting $\operatorname{sgn} y_*$. The expected error is given by

$$e_{\text{Gibbs}}(\boldsymbol{x}_*, t_*) = \Pr_{y_* \sim Q(y_* \,|\, \boldsymbol{x}_*, S)}\{\operatorname{sgn} y_* \neq t_*\} = \Phi\left(\frac{-t_* \mu(\boldsymbol{x}_*)}{\sigma(\boldsymbol{x}_*)}\right), \tag{10}$$

where Φ denotes the cumulative distribution function (c.d.f.) of $N(0,1)$.

Finally note that another frequently used way to introduce Gaussian process models is to view them as linear models with Gaussian prior distributions in a feature space which is typically infinite-dimensional. This view is very useful when designing new algorithms for approximate inference, since many ideas proposed originally for linear models can be imported rather straightforwardly. However, the feature space view requires a mathematically rigorous treatment involving some functional analysis over so-called reproducing kernel Hilbert spaces, while working with Gaussian process models in the way introduced here is completely elementary. In this paper, we do not require the feature space view. We refer to (Williams, 1997) for a discussion of the relationships between the two views, and to (Wahba, 1990, Chap. 1) for the mathematical details required to establish the feature space view.

2.2 The Relative Entropy between Posterior and Prior Gaussian Process

In Section 1.2, we introduced Gaussian processes as distributions over random functions $y(\cdot) : \mathcal{X} \to \mathbb{R}$, and in Section 2.1 we defined two Gaussian processes in particular: the zero-mean prior process P with covariance kernel K and the corresponding Gaussian process approximation Q to the true intractable posterior process, as given by (7) and (8). Our main result is an application of the general PAC-Bayesian Theorem 1 to approximate Bayesian GPC. For this non-parametric model, the parameter vector is the latent function itself, i.e. $\boldsymbol{w} \equiv y(\cdot)$, and the prior P and posterior Q are the corresponding Gaussian process distributions. To this end, we need to compute the Radon-Nikodym derivative $dQ(y(\cdot))/dP(y(\cdot))$ and the relative entropy term (1).

It is easy to see that

$$\frac{dQ(y(\cdot))}{dP(y(\cdot))} = \frac{Q(\boldsymbol{y}_S \,|\, S)}{P(\boldsymbol{y}_S)},$$

where $\boldsymbol{y}_S = y(X_S)$, the outputs over the training input points. Here, $P(\boldsymbol{y}_S) = N(\boldsymbol{y}_S \,|\, \boldsymbol{0}, \boldsymbol{K}_S)$, and $Q(\boldsymbol{y}_S \,|\, S)$ is given by (8). All we need to show is that the process defined by

$$d\hat{Q}(y(\cdot)) = \frac{Q(\boldsymbol{y}_S \,|\, S)}{P(\boldsymbol{y}_S)}\, dP(y(\cdot))$$

is a Gaussian process with the same parameters as Q. To see this, let $X \subset \mathcal{X}$ be finite and define $\boldsymbol{y} = y(X)$. Recall the notations $\boldsymbol{y} \setminus \boldsymbol{y}_S$ and $\boldsymbol{y}_S \setminus \boldsymbol{y}$ from Section 2.1. Then we have

$$\hat{Q}(\boldsymbol{y}) = \int \frac{Q(\boldsymbol{y}_S \,|\, S)}{P(\boldsymbol{y}_S)} P(\boldsymbol{y}, \boldsymbol{y}_S)\, d(\boldsymbol{y}_S \setminus \boldsymbol{y}) = \int Q(\boldsymbol{y}_S \,|\, S) P(\boldsymbol{y} \setminus \boldsymbol{y}_S \,|\, \boldsymbol{y}_S)\, d(\boldsymbol{y}_S \setminus \boldsymbol{y}) = Q(\boldsymbol{y} \,|\, S),$$

where the last equality uses (7). Therefore, the relative entropy $\mathrm{D}[Q \,\|\, P]$ is simply

$$\mathrm{D}[Q \,\|\, P] = \mathrm{E}_{y(\cdot)\sim Q}\left[\log \frac{Q(\boldsymbol{y}_S \,|\, S)}{P(\boldsymbol{y}_S)}\right] = \mathrm{D}[Q(\boldsymbol{y}_S \,|\, S) \,\|\, P(\boldsymbol{y}_S)].$$

Since both $Q(\boldsymbol{y}_S \,|\, S)$ and $P(\boldsymbol{y}_S)$ are Gaussians in $\mathbb{R}^n$, this is easily computed (e.g., Kullback, 1959):

$$\mathrm{D}[Q \,\|\, P] = \frac{1}{2}\log\left|\boldsymbol{\Sigma}_S^{-1}\boldsymbol{K}_S\right| + \frac{1}{2}\operatorname{tr}\left(\boldsymbol{\Sigma}_S^{-1}\boldsymbol{K}_S\right)^{-1} + \frac{1}{2}\hat{\boldsymbol{\alpha}}_S^T\boldsymbol{K}_S\hat{\boldsymbol{\alpha}}_S - \frac{n}{2}. \tag{11}$$

This formula depends of course on the parameters $\hat{\boldsymbol{\alpha}}_S$ and $\boldsymbol{\Sigma}_S$ of the posterior approximation $Q(\boldsymbol{y}_S \,|\, S)$. In Section 3, we show how to compute the relative entropy for a range of concrete GPC approximations.

2.3 Main Result

In this section, we state and discuss our main result, namely an application of the PAC-Bayesian Theorem 1 to Gibbs variants of approximate Bayesian Gaussian process classification methods from the class introduced in Section 2.1. Examples for how this result looks like for several concrete methods are given in Section 3.

Choose some $\delta \in (0,1)$. Then, the following result holds for any zero-mean Gaussian process prior P with covariance kernel K.

Theorem 2 (PAC-Bayesian Theorem for GPC) *For any data distribution over $\mathcal{X} \times \{-1,+1\}$, we have that the following bound holds, where the probability is over random i.i.d. samples $S = \{(\boldsymbol{x}_i^S, t_i^S) \mid i = 1,\dots,n\}$ of size n drawn from the data distribution:*

$$\mathrm{Pr}_S\left\{\mathrm{gen}(Q) > \mathrm{emp}(S,Q) + \mathrm{D}_{\mathrm{Ber}}^{-1}(\mathrm{emp}(S,Q), \varepsilon(\delta,n,P,Q))\right\} \le \delta. \tag{12}$$

Here, we have

$$\begin{aligned}
\mathrm{emp}(S,Q) &= \frac{1}{n}\sum_{i=1}^{n} \mathrm{Pr}_{y_i\sim Q(y_i \,|\, \boldsymbol{x}_i^S, S)}\left\{\operatorname{sgn} y_i \ne t_i^S\right\}, \\
\mathrm{gen}(Q) &= \mathrm{E}_{(\boldsymbol{x}_*, t_*)}\left[\mathrm{Pr}_{y_*\sim Q(y_* \,|\, \boldsymbol{x}_*, S)}\left\{\operatorname{sgn} y_* \ne t_*\right\}\right], \\
\varepsilon(\delta,n,P,Q) &= \frac{1}{n}\left(\mathrm{D}[Q \,\|\, P] + \log\frac{n+1}{\delta}\right).
\end{aligned}$$

Thus, $\mathrm{emp}(S,Q)$ *is the expected empirical error,* $\mathrm{gen}(Q)$ *the expected generalisation error of the GP Gibbs classifier (note that the probability in* $\mathrm{gen}(Q)$ *is over $(\boldsymbol{x}_*, t_*)$ drawn from the data distribution, independently from the sample S) whose predictive distribution $Q(y_* \,|\, \boldsymbol{x}_*, S)$ is given by (9), and* $\mathrm{D}_{\mathrm{Ber}}^{-1}(q,\varepsilon)$ *is defined by (4). Finally,* $\mathrm{D}[Q \,\|\, P]$ *in $\varepsilon(\delta,n,P,Q)$ is given in (11). All of these terms depend on the parameters $\hat{\boldsymbol{\alpha}}_S$, $\boldsymbol{\Sigma}_S$ of the posterior approximation (8).*

We have essentially already proved this theorem. In Section 2.1, we have shown how to compute the predictive distribution $Q(y_* \,|\, \boldsymbol{x}_*, S)$ (see Equation 9), thus $\mathrm{emp}(S,Q)$ can be computed easily using (10). $\varepsilon(\delta,n,P,Q)$ is computed using (11). Below, when specialising to several concrete methods (i.e. "fill in" $\hat{\boldsymbol{\alpha}}_S$ and $\boldsymbol{\Sigma}_S$), we will give more detailed comments on how to compute the terms the bound depends upon.

3. Applications to Concrete Gaussian Process Classification Methods

In the previous section, we stated and proved our main result (Theorem 2) which is valid for a large class of approximate Bayesian GPC techniques. In this section, we will instantiate this result with two particular GPC methods, both of high practical relevance. For these methods, which will both be introduced briefly, we provide computational details and some further analysis. The experiments presented in Section 4 are based on the GPC methods we describe here.

3.1 Laplace Gaussian Process Classification

In this section, we concentrate on a particular simple, yet powerful approximate GPC method suggested by Williams and Barber (1998). This technique is referred to as *Laplace Gaussian process classification*, and we will begin by briefly introducing this framework. A detailed introduction can be found in (Williams and Barber, 1998).

Recall from Section 1.1 that we are given some i.i.d. data sample S of size n, drawn from an unknown data distribution. Our noise model will be Bernoulli (logit), i.e. $P(t|y) = \sigma(ty)$, and for this non-Gaussian noise distribution, exact Bayesian analysis is intractable. In a nutshell, the Laplace GPC approximation works by first determining the vector $\hat{\boldsymbol{y}}_S$ maximising the posterior $P(\boldsymbol{y}_S|S)$, where $\boldsymbol{y}_S = y(X_S)$, $X_S = \{\boldsymbol{x}_1^S, \dots, \boldsymbol{x}_n^S\}$. This is a convex optimisation problem, and therefore has a unique solution. Let $\boldsymbol{K}_S = \boldsymbol{K}(X_S)$. In our context, it is useful to operate on the dual vector $\boldsymbol{\alpha}_S = \boldsymbol{K}_S^{-1}\boldsymbol{y}_S$. Then, the log posterior can be written, up to additive constants, as a criterion

$$F(\boldsymbol{\alpha}_S, \boldsymbol{K}_S) = \sum_{i=1}^{n} \log \sigma(t_i^S y_i^S) - \frac{1}{2}\boldsymbol{\alpha}_S^T \boldsymbol{K}_S \boldsymbol{\alpha}_S, \quad \boldsymbol{y}_S = (y_i^S)_i = \boldsymbol{K}_S\boldsymbol{\alpha}_S. \tag{13}$$

Since F is concave, it has a unique maximiser $\hat{\boldsymbol{\alpha}}_S$ which can be found using the Newton-Raphson algorithm. Furthermore, $\hat{\boldsymbol{y}}_S = \boldsymbol{K}_S\hat{\boldsymbol{\alpha}}_S$ is the posterior mode. We now approximate the posterior $P(\boldsymbol{y}_S|S)$ by a Gaussian $Q(\boldsymbol{y}_S|S)$, using the Laplace approximation (e.g., Kaas and Raftery, 1995) around the mode $\hat{\boldsymbol{y}}_S$. This results in

$$Q(\boldsymbol{y}_S|S) = N(\boldsymbol{y}_S|\boldsymbol{K}_S\hat{\boldsymbol{\alpha}}_S, (\boldsymbol{W} + \boldsymbol{K}_S^{-1})^{-1}), \tag{14}$$

where $\boldsymbol{W}$ is a diagonal matrix with positive entries. In fact, at the mode $\hat{\boldsymbol{y}}_S$, we have:

$$\hat{\boldsymbol{\alpha}}_S = (t_i^S \sigma(-t_i^S \hat{y}_i^S))_i, \quad \boldsymbol{W} = \operatorname{diag}(\sigma(-t_i^S \hat{y}_i^S)\sigma(t_i^S \hat{y}_i^S))_i. \tag{15}$$

Note that if $\boldsymbol{\Sigma}_S = (\boldsymbol{W} + \boldsymbol{K}_S^{-1})^{-1}$, then (14) becomes consistent with (8). If we define the positive definite matrix

$$\boldsymbol{A} = \boldsymbol{I}_n + \boldsymbol{W}^{1/2}\boldsymbol{K}_S\boldsymbol{W}^{1/2}, \tag{16}$$

then some matrix algebra results in

$$\begin{aligned} \boldsymbol{M}_S = \boldsymbol{K}_S^{-1} - \boldsymbol{K}_S^{-1}\boldsymbol{\Sigma}_S\boldsymbol{K}_S^{-1} &= \boldsymbol{W}^{1/2}\boldsymbol{A}^{-1}\boldsymbol{W}^{1/2}, \\ \boldsymbol{\Sigma}_S^{-1}\boldsymbol{K}_S &= \boldsymbol{W}^{1/2}\boldsymbol{A}\boldsymbol{W}^{-1/2}. \end{aligned} \tag{17}$$

This allows us to compute the predictive distribution (9) and the relative entropy term (11) in a stable way. Suppose we are given the Cholesky decomposition (e.g., Horn and Johnson, 1985, Chap. 7) $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T$, where $\boldsymbol{L}$ is lower-triangular with positive diagonal. Then, the predictive variance is computed as

$$\sigma^2(\boldsymbol{x}_*) = K(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{r}^T\boldsymbol{r}, \quad \boldsymbol{L}\boldsymbol{r} = \boldsymbol{W}^{1/2}\boldsymbol{k}(\boldsymbol{x}_*),$$

which is $O(n^2)$ due to the back-substitution for $\boldsymbol{r}$. The Cholesky decomposition is by far the most stable (and also most efficient) exact method to do these computations, and we discourage the reader from using other techniques involving matrix inversions. The evaluation of the expected empirical error $\mathrm{emp}(S, Q)$ in Theorem 2 requires the evaluation of the predictive variances at the training points, thus is $O(n^3)$.

Using (17), the relative entropy term (11) simplifies to

$$\mathrm{D}[Q \,\|\, P] = \frac{1}{2}\log|\boldsymbol{A}| + \frac{1}{2}\operatorname{tr}\boldsymbol{A}^{-1} + \frac{1}{2}\hat{\boldsymbol{\alpha}}_S^T\boldsymbol{K}_S\hat{\boldsymbol{\alpha}}_S - \frac{n}{2}. \tag{18}$$

Note that $\log|\boldsymbol{A}| = 2\log|\operatorname{diag}\boldsymbol{L}|$. The term $\operatorname{tr}\boldsymbol{A}^{-1}$ can be computed from $\boldsymbol{L}$ in roughly the same time as $\boldsymbol{L}$ is obtained from $\boldsymbol{A}$. Thus, the computation of (18) is $O(n^3)$ as well.

The Gibbs classifier seems unattractive for predictions on large test sets, because each evaluation requires $O(n^2)$. In contrast to this, the Bayes classifier variant depends on the mean $\mu(\boldsymbol{x}_*) = \boldsymbol{k}(\boldsymbol{x}_*)^T\hat{\boldsymbol{\alpha}}_S$ only, which is $O(n)$ per test point. However, we show in (Seeger, 2002) how one can use sparse approximation techniques together with rejection sampling in order to avoid the exact computation of the variance term for most of the predictions, ending up with $O(n)$ (average case) per prediction.

3.1.1 Some Analysis of the Relative Entropy Term

In this section, we present some analysis of the relative entropy term $\mathrm{D}[Q \,\|\, P]$ (see Equation 18) in the bound of Theorem 2, as applied to Laplace GPC. In normal situations (i.e. δ not extremely small), the expression $\varepsilon(\delta, n, P, Q)$ in (12) is dominated by this term.

Now to the remaining terms in (18). The matrix $\boldsymbol{A}$ of (16) is positive definite, and all its eigenvalues are ≥ 1. Furthermore, by taking any unit vector $\boldsymbol{u}$ and using the min-max characterisation of the eigenvalue spectrum (e.g., Horn and Johnson, 1985, Sect. 4.2) of Hermitian matrices, we have $\boldsymbol{u}^T\boldsymbol{A}\boldsymbol{u} = 1 + (\boldsymbol{W}^{1/2}\boldsymbol{u})^T\boldsymbol{K}_S(\boldsymbol{W}^{1/2}\boldsymbol{u}) \leq 1 + \lambda_{\max}\boldsymbol{u}^T\boldsymbol{W}\boldsymbol{u} < 1 + \lambda_{\max}/4$, where $\lambda_{\max}$ is the largest eigenvalue of $\boldsymbol{K}_S$ (note that the positive coefficients of $\boldsymbol{W}$ are all $< 1/4$). Thus, all eigenvalues of $\boldsymbol{A}$ lie in $(1, 1 + \lambda_{\max}/4)$. By analysing $(1/2)\log|\boldsymbol{A}| + (1/2)\operatorname{tr}\boldsymbol{A}^{-1}$ for general[8] $\boldsymbol{A} \succeq \boldsymbol{I}_n$, we see that this term must lie between $n/2$ and $(n/2)(\log(1 + \lambda_{\max}/4) + (1 + \lambda_{\max}/4)^{-1})$, although these bounds are not necessarily tight.

3.2 Sparse Greedy Gaussian Process Classification

A principal drawback of many approximate GPC techniques in practice is their scaling of $O(n^3)$ with the sample size n during training. For example, the Laplace GPC technique discussed in Section 3.1, although one of the fastest (non-sparse) known GPC techniques, scales as $O(n^3)$ in a straightforward implementation.[9] Furthermore, these techniques typ-

8. $\boldsymbol{A} \succeq \boldsymbol{I}_n$ means that $\boldsymbol{A} - \boldsymbol{I}_n$ is positive semidefinite.
9. This is true for Laplace Gibbs GPC and for the evaluation of the terms determining the bound value in Theorem 2. Laplace Bayes GPC typically scales as $O(n^2)$ (average case), if the Newton steps are approximated using a conjugate gradients solver.

First, we can use (15) to show that $\hat{\boldsymbol{\alpha}}_S^T \boldsymbol{K}_S \hat{\boldsymbol{\alpha}}_S = \hat{\boldsymbol{y}}_S^T \hat{\boldsymbol{\alpha}}_S = \sum_i t_i^S \hat{y}_i^S \sigma(-t_i^S \hat{y}_i^S)$, which is simply $\sum_i f(t_i^S \hat{y}_i^S)$, $f(x) = x\,\sigma(-x)$, and $t_i^S \hat{y}_i^S$ is the so-called *margin* at example $(\boldsymbol{x}_i^S, t_i^S)$. $f(x)$ is plotted in Figure 1. It is maximal at $x_* \approx 1.28$, converges to 0 exponentially quickly for $x \to \infty$ and behaves like $x \mapsto x$ for $x \to -\infty$. Thus, at least w.r.t. the third term in (18), classification mistakes (i.e. $t_i^S \hat{y}_i^S < 0$) render a negative contribution to the gap bound value. This is what we expect for a Bayesian architecture. Namely, the method chose a *simple* solution, at the expense of making this mistake, yet with the goal to prevent disastrous over-fitting. In our bound, we are penalised by a higher empirical error, but we should be rewarded by a smaller gap bound term.

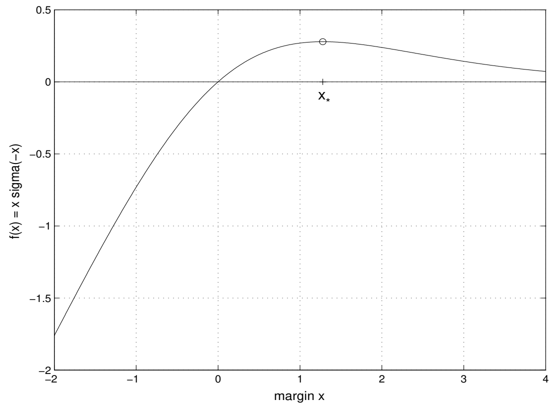

Figure 1: Relation between margin and gap bound part

ically require the evaluation of the complete kernel matrix $\boldsymbol{K}_S$. Recently, a number of *sparse* approximation techniques for Bayesian GPC have been proposed (e.g., Tipping, 2001, Williams and Seeger, 2001, Smola and Bartlett, 2001, Tresp, 2000, Csató and Opper, 2002, Lawrence, Seeger, and Herbrich, 2002), providing an important alternative for practitioners. The common theme of these techniques is to restrict the number of coefficients to be used in the final discriminant expansion to a controllable number $k \ll n$ (an exception is the method of Williams and Seeger (2001) which results in a dense expansion). By means of this, some of these methods achieve a training complexity of at most $O(nk^2)$. This involves the selection of a "representative" subset of size k of the training inputs. Since an optimal selection (in any meaningful sense) is intractable, the methods resort to randomised and/or greedy strategies, often of heuristic nature.

Here, we focus on a class of sparse GPC techniques proposed by Csató and Opper (2002) and Lawrence, Seeger, and Herbrich (2002). Due to space limitations, a detailed description of these methods cannot be given here, for details see (Seeger et al., 2002). In what follows, we introduce aspects of the methods we require in this section, but the exposition is not self-contained. We then show how the terms determining the bound value of Theorem 2 can be evaluated in time $O(nk^2)$.

The method falls in the class described in Section 2.1 and propagates Gaussian approximations $Q(\boldsymbol{y}_S|S)$ to the intractable true posterior $P(\boldsymbol{y}_S|S)$ (see Csató and Opper, 2002, Lawrence, Seeger, and Herbrich, 2002). $Q(\boldsymbol{y}_S|S)$ is parameterised by a diagonal matrix $\boldsymbol{\Pi}$ with nonnegative elements and a vector $\boldsymbol{m}$:

$$Q(\boldsymbol{y}_S|S) = N\left(\boldsymbol{y}_S | \boldsymbol{K}_S(\boldsymbol{I}_n + \boldsymbol{\Pi}\boldsymbol{K}_S)^{-1}\boldsymbol{T}\boldsymbol{\Pi}\boldsymbol{m}, (\boldsymbol{K}_S^{-1} + \boldsymbol{\Pi})^{-1}\right),$$

where $\boldsymbol{T} = \operatorname{diag}(t_i^S)_i$ contains the targets. This becomes consistent with (8) if we set $\hat{\boldsymbol{\alpha}}_S = (\boldsymbol{I}_n + \boldsymbol{\Pi}\boldsymbol{K}_S)^{-1}\boldsymbol{T}\boldsymbol{\Pi}\boldsymbol{m}$ and $\boldsymbol{\Sigma}_S = (\boldsymbol{K}_S^{-1} + \boldsymbol{\Pi})^{-1}$. In *sparse* variants of this approximation, we allow for elements of $\operatorname{diag}\boldsymbol{\Pi}$ to be zero. In fact, for such methods we keep an *active* *set* $I \subset \{1, \dots, n\}$, with $k = |I| \ll n$, and make sure that if $\operatorname{diag} \boldsymbol{\Pi} = (p_i)_i$, then $p_i = 0$ whenever $i \notin I$. The active set is grown up to a desired size (or until a stopping criterion is met) by *including* new datapoints $(\boldsymbol{x}_i^S, t_i^S)$. In order to include a new point i, the algorithm only requires the i-th row of the kernel matrix $\boldsymbol{K}_S$, i.e. $\boldsymbol{K}(\{\boldsymbol{x}_i^S\}, X_S)$. After I has reached the desired size, the method can be stopped, but some variants try to continue to refine the approximation while keeping the size of I constant. In *sparse greedy* variants of this scheme, the next datapoint to be included is selected greedily using a heuristic scoring criterion. In this work, we employ the heuristic proposed in (Lawrence, Seeger, and Herbrich, 2002), referred to as *differential entropy score*, which can be evaluated very efficiently.[10] The first few patterns to be included are chosen at random. Later, we select a pattern by scoring *all* remaining ones using the differential entropy criterion and pick the winner. The algorithm stops once k patterns have been included.

Due to the fact that at most k of the elements in $\operatorname{diag} \boldsymbol{\Pi}$ are non-zero, it turns out that the quantities depending on the final posterior approximation $Q(\boldsymbol{y}_S|S)$ which we are interested in, namely the parameters of the predictive distribution (9) and the relative entropy term (11) can be computed efficiently and using only the part $[\boldsymbol{K}_S]_{I,\cdot}$ of the kernel matrix. First note that

$$\boldsymbol{M}_S = (\boldsymbol{I}_n + \boldsymbol{\Pi}\boldsymbol{K}_S)^{-1} \boldsymbol{\Pi}, \quad \boldsymbol{\Sigma}_S^{-1}\boldsymbol{K}_S = \boldsymbol{I}_n + \boldsymbol{\Pi}\boldsymbol{K}_S.$$

Denote $\boldsymbol{\Pi}_k = [\boldsymbol{\Pi}]_{I,I}$ and define

$$\boldsymbol{B} = \boldsymbol{I}_k + \boldsymbol{\Pi}_k^{1/2}[\boldsymbol{K}_S]_{I,I}\boldsymbol{\Pi}_k^{1/2} \in \mathbb{R}^{k \times k}, \quad \boldsymbol{\beta} = \boldsymbol{\Pi}_k^{1/2}\boldsymbol{B}^{-1}\boldsymbol{\Pi}_k^{1/2} [\boldsymbol{T}\boldsymbol{m}]_I \in \mathbb{R}^k.$$

If $\boldsymbol{E}_I \in \mathbb{R}^{k,n}$ denotes the "selection" matrix for I, i.e. $\boldsymbol{E}_I\boldsymbol{g} = [\boldsymbol{g}]_I$, then some elementary matrix algebra using the Sherman-Morrison-Woodbury formula (e.g., Press et al., 1992, Sect. 2.7) gives

$$\begin{aligned} (\boldsymbol{I}_n + \boldsymbol{\Pi}\boldsymbol{K}_S)^{-1} &= \boldsymbol{I}_n - \boldsymbol{E}_I^T\boldsymbol{\Pi}_k^{1/2}\boldsymbol{B}^{-1}\boldsymbol{\Pi}_k^{1/2}[\boldsymbol{K}_S]_{I,I}\boldsymbol{E}_I, \\ \boldsymbol{M}_S &= \left(\boldsymbol{\Pi}_k^{1/2}\boldsymbol{E}_I\right)^T \boldsymbol{B}^{-1}\boldsymbol{\Pi}_k^{1/2}\boldsymbol{E}_I. \end{aligned} \tag{19}$$

Since $\hat{\boldsymbol{\alpha}}_S = \boldsymbol{M}_S\boldsymbol{T}\boldsymbol{m}$, we have $\hat{\boldsymbol{\alpha}}_S = \boldsymbol{E}_I^T\boldsymbol{\beta}$. The predictive variance is best computed using the Cholesky decomposition $\boldsymbol{B} = \boldsymbol{L}_k\boldsymbol{L}_k^T$. Plugging (19) into (9), we have

$$\mu(\boldsymbol{x}_*) = \boldsymbol{\beta}^T\boldsymbol{k}_I(\boldsymbol{x}_*), \sigma^2(\boldsymbol{x}_*) = K(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{r}^T\boldsymbol{r}, \quad \boldsymbol{L}_k\boldsymbol{r} = \boldsymbol{\Pi}_k^{1/2}\boldsymbol{k}_I(\boldsymbol{x}_*),$$

where $\boldsymbol{k}_I(\boldsymbol{x}_*) = (K(\boldsymbol{x}_*, \boldsymbol{x}_i^S))_{i \in I} \in \mathbb{R}^k$. Thus, each Gibbs prediction can be computed in $O(k^2)$, and the expected empirical error of the Gibbs classifier is obtained at a cost of

10. We remark that although the schemes proposed in (Csató and Opper, 2002) and (Lawrence, Seeger, and Herbrich, 2002) seem quite similar, in that they include new datapoints into the approximation in the same way, they are algorithmically quite different. The method proposed by Csató and Opper (2002) is an on-line scheme in which datapoints are used to refine the posterior approximation *even if* they are not included into the active set, while this does not hold for the method suggested in (Lawrence et al., 2002). This means that the latter can be significantly faster, yet less accurate in practice, although both methods scale as $O(nk^2)$. Furthermore, the greedy method is a compression scheme (see Section 4.3), which is not the case for the on-line algorithm.

$O(nk^2)$. Note that the corresponding Bayes classifier can be evaluated in $O(k)$, due to the sparse expansion for $\mu(\boldsymbol{x}_*)$. For the computation of the relative entropy term (11), we use (19) and the well-known matrix formulas $|\boldsymbol{I}+\boldsymbol{U}\boldsymbol{V}| = |\boldsymbol{I}+\boldsymbol{V}\boldsymbol{U}|$ and $\operatorname{tr}\boldsymbol{U}\boldsymbol{V} = \operatorname{tr}\boldsymbol{V}\boldsymbol{U}$ (the former follows from a well-known formula using Schur products, e.g., Horn and Johnson, 1985, Sect. 0.8.5). Then, $\log|\boldsymbol{\Sigma}_S^{-1}\boldsymbol{K}_S| = \log|\boldsymbol{B}|$ and $\operatorname{tr}(\boldsymbol{\Sigma}_S^{-1}\boldsymbol{K}_S)^{-1} = n-k+\operatorname{tr}\boldsymbol{B}^{-1}$, therefore

$$\mathrm{D}[Q\,\|\,P] = \frac{1}{2}\log|\boldsymbol{B}| + \frac{1}{2}\operatorname{tr}\boldsymbol{B}^{-1} + \frac{1}{2}\boldsymbol{\beta}^T\left[\boldsymbol{K}_S\right]_{I,I}\boldsymbol{\beta} - \frac{k}{2}. \tag{20}$$

This is computed in $O(k^3)$, given the Cholesky decomposition of $\boldsymbol{B}$. All in all, the upper bound on $\mathrm{gen}(Q)$ given by Theorem 2 for sparse greedy GPC can be computed in $O(nk^2)$.

It is interesting to point out the close similarity between the two relative entropy formulas (18) and (20). This is due to the similar form of $\boldsymbol{\Sigma}_S$ (covariance matrix of $Q(\boldsymbol{y}_S|S)$) both methods are employing, namely $\boldsymbol{\Sigma}_S = (\boldsymbol{K}_S^{-1}+\boldsymbol{D})^{-1}$, where $\boldsymbol{D}$ is a diagonal matrix. In the Laplace GPC case, with $\boldsymbol{D}=\boldsymbol{W}$, the components of $\operatorname{diag}\boldsymbol{D}$ are positive, and there is no force which drives many of these components towards zero. In the sparse greedy GPC case, with $\boldsymbol{D}=\boldsymbol{\Pi}$, $n-k$ of the components of $\operatorname{diag}\boldsymbol{D}$ are zero, allowing us to use more efficient computations. Note that the method we have discussed in this section becomes identical to the "cavity" approach suggested by Opper and Winther (2000) if we let $k=n$, i.e. allow all components of $\boldsymbol{\Pi}$ to become non-zero (see also Minka, 2001). Another idea to control sparsity in $\boldsymbol{\Pi}$, different from the approaches in (Csató and Opper, 2002, Lawrence, Seeger, and Herbrich, 2002), would be to place special priors on the components of $\boldsymbol{\Pi}$, forcing them to be small under the posterior if the data does not suggest otherwise. This is the approach adopted by the *relevance vector machine (RVM)* of Tipping (2001), although there the sparsity of a set of parameters different from $\boldsymbol{\Pi}$ is controlled.

We would also like to remark that our bound for sparse greedy GP methods is (in general) not a compression bound. Theorems of the latter class require that the finally selected discriminant is exactly recovered if we restrict ourselves, in an independent training run, to the datapoints in the final expansion as a training set (see Section 4.3). Although our bound applies to compression scheme versions of sparse greedy GPC, it is not restricted to such. The experiments presented in Sections 4.2 and 4.3 suggest that Theorem 2 for sparse greedy GPC renders a much tighter generalisation error bound than a standard PAC compression bound in situations where both bounds apply.

4. Experiments

Here, we present experiments testing our main result (Theorem 2) for the Laplace GP Gibbs classifier of Section 4.1 and the sparse greedy GP Gibbs classifier of Section 4.2, using a setup to be described shortly. The results indicate that the bounds are very tight even for training samples of moderate sizes. In Section 4.3, we compare our bound to a state-of-the-art PAC compression bound for the sparse greedy GP Bayes classifier, and to the same compression bound for the soft-margin support vector classifier in Section 4.3.1. Finally, in Section 4.4 we try to evaluate the model selection qualities of our result for sparse greedy GP classification.

A real-world binary classification task was created on the basis of the well-known MNIST handwritten digits database[11] as follows. MNIST comes with a training set of 60000 and a test set of 10000 handwritten digits, represented as 28-by-28-pixel bitmaps, the pixel intensities are quantised to 8 bit values. First, the input dimensionality was reduced by cutting away a 2-pixel margin, then averaging intensities over 3-by-3-pixel blocks, resulting in 8-by-8-pixel bitmaps. The task of discriminating handwritten twos against threes is among the harder binary ones.[12] By selecting these digits only, a training pool of 12089 cases and a test set of $l = 1000$ cases were created.

For our experiments, we employed the frequently used *Radial Basis Functions (RBF)* covariance kernel

$$K(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}) = C \exp\left(-\frac{w}{2d} \left\| \boldsymbol{x}^{(1)} - \boldsymbol{x}^{(2)} \right\|^2 \right).$$

Here, d is the dimensionality of the inputs ($d = 64$ in our case), w and C are positive parameters. C determines the variance of the underlying random process (see Section 1.2), while $w^{-1/2}$ determines its average length scale.

The experimental setup is as follows. An experiment consists of $L = 10$ independent iterations. During an iteration, three datasets are sampled independently and without replacement from the training pool: a model selection (MS) training set of size n_{MS}, a MS validation set of size l_{MS} and a task training sample S of size n. Note that the latter set is sampled independently from the model selection sets, ensuring that the prior P in Theorem 2 is independent of the task training sample. This issue is discussed in more detail in Section 5. Then, model selection is performed over a list of candidates for (w, C), where a classifier is trained on the MS training set and evaluated on the MS validation set (the MS score is the expected empirical error of the Gibbs classifier on the MS validation set). The winner is then trained on the task training set and evaluated on the test set. Alongside, the upper bound value given by Theorem 2 is evaluated, where the confidence parameter δ is fixed to 0.01. We also quote total running time, as observed on a DEC Alpha workstation with almost four gigabytes of RAM.

4.1 Experiments with Laplace GPC

Our implementation uses the Newton-Raphson algorithm in order to maximise the log posterior criterion (13). The Newton steps are computed using a conjugate gradients solver for symmetric positive definite linear systems. The prediction vector $\hat{\boldsymbol{\alpha}}_S$ is found in $O(n^2)$ (average case). The Cholesky decomposition of the system matrix (16), the evaluation of the expected empirical error of the Gibbs classifier and of the relative entropy term (18) require $O(n^3)$ each. The specifications and results for the experiments of this section are listed in Table 1. For all these experiments, we chose model selection validation set size $l_{\mathrm{MS}} = 1000$ (recall that the test set is fixed with size $l = 1000$). Experiments #1 to #5 have growing sample sizes $n = 500, 1000, 2000, 5000, 9000$, the corresponding MS training set sizes are $n_{\mathrm{MS}} = 1000$ for experiments #2 to #5, and $n_{\mathrm{MS}} = 500$ for experiment #1. Note that $n_{\mathrm{MS}} < n$ in experiments #3 to #5 is chosen for computational feasibility, due to the

11. Available online at *http://www.research.att.com/~yann/exdb/mnist/index.html*.
12. We will consider other binary subtasks of MNIST as well as other binary tasks in future work.

considerable size of the candidate list for (C, w). In Table 2, we list additional information about the experiments.

#	n	n_{MS}	emp	gen	upper
1	500	500	0.036	0.0469	0.182
			($\pm$0.0039)	($\pm$0.0015)	($\pm$0.0057)
2	1000	1000	0.0273	0.036	0.131
			($\pm$0.0023)	($\pm$0.001)	($\pm$0.0041)
3	2000	1000	0.0243	0.028	0.1091
			($\pm$0.0026)	($\pm$0.0013)	($\pm$0.0079)
4	5000	1000	0.0187	0.0195	0.076
			($\pm$0.0016)	($\pm$0.0011)	($\pm$0.002)
5	9000	1000	0.0178	0.0172	0.0706
			($\pm$0.0012)	($\pm$0.0013)	($\pm$0.0037)

Table 1: Experimental results for Laplace GPC. n: task training set size; n_{MS}: model selection training set size. emp: expected empirical error; gen: expected generalisation error (estimated as average over test set). upper: upper bound on expected generalisation error after Theorem 2. Figures are mean and width of 95% t-test confidence interval.

#	gen-bayes	C pars	w pars	approx-time
1	0.0339	50(6),30(2),20,25	0.5(5),2(3),0.75(2)	14 min
	($\pm$0.0023)			
2	0.0274	10(9),20	3(5),2(2),5	67 min
	($\pm$0.0022)			
3	0.0236	5(5),3(3),10(2)	10(6),7.5(2),5(2)	91 min
	($\pm$0.0029)			
4	0.0171	5(6),7.5(2),15,20	7.5(3),10(3),12(2),5,3	762 min
	($\pm$0.0016)			
5	0.0158	2(4),3(2),5(2),7.5(2)	12(4),10(3),5(2),7.5	3618 min
	($\pm$0.0017)			

Table 2: Additional information for Laplace GPC experiments. gen-bayes: test error of corr. Bayes classifier. C pars, w pars: Values of C, w chosen by MS (frequency in parentheses). approx-time: approximate total running time. Figures are mean and width of 95% t-test confidence interval.

Note that the resource requirements for our experiments are well within today's desktop machines computational capabilities. For example, experiment #4 was completed in total time of about 12 to 13 hours, the memory requirements are around 250M. Now, for this setting both the expected empirical error and the estimate (on the test set) of the expected generalisation error lie around 2%, while the PAC bound on the expected generalisation error given by Theorem 2 is 7.6% — an impressive, highly nontrivial result on samples of size

$n = 5000$. Our largest experiment #5 was done mainly for comparison with experiment #2 for sparse greedy GPC (see Section 4.2). The total computation time was 6 hours for each iteration, and the memory requirements are around 690M. We note a slight improvement in test errors as well as in the upper bound values (which now lie around 7%).

The "gen-bayes" column in Table 2 contains the test error that a Bayes classifier with the same approximate posterior as the Gibbs classifier attains. Note that it is not necessarily the best we could obtain for a Bayes classifier, because the model selection is done specifically for the Gibbs, not the Bayes classifier. In the Laplace GPC case we note that Bayes and Gibbs variants perform comparably well, although the Bayes classifier attains slightly better results and, as mentioned in Section 2.1, can be evaluated more efficiently. We include these results for comparison only: although our main result implies a bound on the generalisation error of the Bayes classifier (see Section 1.3.1), the link is too weak to render a sufficiently tight result.

4.2 Experiments with Sparse Greedy GPC

The algorithmic details of our implementation can be found in a separate paper (Seeger et al., 2002). Training proceeds in two phases. In the first phase, a number k_{rand} of patterns from the training sample are selected at random and included into the approximation. In the second phase, we include further patterns until the active set has grown to size k. However, now the next pattern to be included is determined by scoring all remaining ones using the differential entropy criterion (see Section 3.2) and choosing the one with the best value. Empirically, we found the greedy selection to be very effective, thus typically $k_{\text{rand}} \ll k$. Note that we require $k_{\text{rand}} \geq 1$, because the differential entropy criterion is constant over all patterns if the active set is empty and the kernel diagonal is constant. We use different values for k and k_{rand} during model selection, denoted by k_{MS}, $k_{\text{rand,MS}}$. For all experiments reported here, we chose MS training size $n_{\text{MS}} = 1000$, MS validation size $l_{\text{MS}} = 1000$, $k_{\text{MS}} = 150$, $k_{\text{rand}} = 3$ and $k_{\text{rand,MS}} = 2$. Note that in experiments which have the same $(n, n_{\text{MS}}, l_{\text{MS}})$ constellation as Laplace GPC experiments, we use the same data subsets, in order to facilitate direct comparisons. The results are listed in Table 3. In Table 4, we list additional information about the experiments.

#	n	k	emp	gen	upper
1	5000	500	0.0154	0.0207	0.067
			(±0.0021)	(±0.0015)	(±0.0026)
2	9000	900	0.0101	0.0116	0.0502
			(± 6.88e-4)	(± 5.49e-4)	(± 6.13e-4)

Table 3: Experimental results for sparse GPC. n: task training set size; k: final active set size. emp: expected empirical error; gen: expected generalisation error (estimated as average over test set). upper: upper bound on expected generalisation error after Theorem 2. Figures are mean and width of 95% t-test confidence interval.

Let us compare these results to the ones obtained for Laplace GPC. The sparse GPC Gibbs classifier trained with 5000 examples attains an expected test error of 2.1%, and the

#	gen-bayes	C pars	w pars	approx-time
1	0.0084 ($\pm$0.0014)	120(9),100(1)	3(5),2(3),5(2)	16 min
2	0.0042 ($\pm$ 8.79e-4)	150(5),125(2),120(2),100	3(7),2(2),5	82 min

Table 4: Additional information for sparse GPC experiments. gen-bayes: test error of corr. Bayes classifier. C pars, w pars: Values of C, w chosen by MS (frequency in parentheses). approx-time: approximate total running time. Figures are mean and width of 95% t-test confidence interval.

upper bound evaluates to 6.7%. While the former is the same as for the Laplace GPC variant, the latter is significantly lower. The ratio between upper bound and expected test error is 3.19, the ratio between gap bound and expected test error is 2.46 — demonstrating an impressive tightness for a state-of-the-art classifier trained on just 5000 examples. Also important for the practitioner, experiment #1 for the sparse GPC was completed in total time of about 16 minutes on the machine mentioned in Section 4.1 — almost fifty times faster than the Laplace GPC experiment #4. Note that the limited size of the task database does not allow samples sizes much larger than $n = 9000$ (experiments on much larger datasets are subject to future work). It is interesting to observe that for this sample size, the results here are significantly better than for the full Laplace GPC on the same task[13] (experiment #5 in Section 4.1). Finally note that we did not try to optimise the final active set size k, but simply fixed $k = n/10$ *a priori*. An automatic choice of k could be based on heuristics which evaluate the error on the datapoints outside the active set.

The “gen-bayes” column in Table 4 serves the same purpose as the “gen-bayes” column in Table 2. In case of sparse greedy GPC, the results show that the Bayes classifier performs somewhat significantly better than the Gibbs variant, although the latter still attains very competitive results. A possible explanation for this difference, given that it cannot be observed for Laplace GPC, is obtained by inspecting the (C, w) kernel parameters values that are preferred by sparse greedy GPC. The parameter C is much larger for sparse GPC, i.e. the latent process has a larger *a priori* variance. This is sensible, because sparse GPC has to rely on much fewer terms in the final expansion than Laplace GPC, thus has to make sure that the kernels corresponding to active patterns span a larger range, and also the coefficients in the expansion (the coefficients of $\boldsymbol{\beta}$) lie in a broader interval. However, this typically leads to an increase in the predictive variances, which in turn might introduce more sampling errors in the Gibbs predictions.

13. Note that we are comparing two entirely different ways of approximating the true posterior by a Gaussian: a Laplace approximation around the *mode* (which is different from the posterior *mean* — the “holy grail” of Bayesian logistic regression, see Section 2.1) and an approximation based on repeated moment matching. A more meaningful direct comparison would involve the TAP method of Opper and Winther (2000) which is, however, significantly more costly to compute than the Laplace approximation.

4.3 Comparison with PAC Compression Bound

In this section, we present further experiments in order to compare our result for sparse GPC with a state-of-the-art *PAC compression bound*. Note that here, we employ *Bayes* GP classifiers instead of *Gibbs* GP classifiers: it would not be fair to compare our Gibbs-specific bound to an "artificially Gibbs-ified" version of a result which is typically used with Bayes classifiers. A compression bound applies to learning algorithms which have a particular characteristic. Namely, suppose we are given a learning algorithm A which maps data samples S of size n to hypotheses $A(S)$, which are then used to classify future input points. A is called a *compression scheme* if there exists another algorithm R, mapping samples of size smaller than n to hypotheses, such that for every sample S we can find a $k < n$ and a subsample of S of size k such that R trained on this subsample outputs the same hypothesis as A trained on S. Popular examples of compression schemes are the perceptron learning algorithm of Rosenblatt (1958) and the support vector machine (see Section 4.3.1).

It turns out that the sparse greedy GPC variant we are using in the experiments reported in Section 4.2, is a compression scheme where k is fixed *a priori*. Herbrich (2001, Theorem 5.18) gives a PAC bound for compression schemes (drawing from earlier work of Littlestone and Warmuth, 1986) which can be considered state-of-the-art. In order to ensure a fair comparison, we use a refined version of this bound which can be found in (Seeger, 2002). There, it is also shown why and to what extent our sparse greedy GPC variant is a compression scheme. The PAC upper bound on the generalisation error depends only on the training error on the remaining $n-k$ patterns of S outside the active set (called $\mathrm{emp}^{\backslash k}(S)$), furthermore on k and k_{rand}. We repeated the experimental setup used in Section 4.2 and employed the same dataset splits. The results can be found in Table 5.

#	n	k	emp	gen	upper
1	5000	500	0.0025	0.0058	0.3048
			(± 6.79e-4)	(±0.0015)	(± 0)
2	9000	900	0.0024	0.003	0.3041
			(± 4.25e-4)	(± 7.54e-4)	(± 0)

Table 5: Experimental results for PAC compression bound with sparse GP Bayes classifier. n: task training set size; k: final active set size. emp: empirical error (on full training set); gen: error on test set. upper: upper bound on generalisation error given by PAC compression bound. Figures are mean and width of 95% t-test confidence interval.

For both experiments, $\mathrm{emp}^{\backslash k}(S) = 0$ was achieved in all runs, the compression bound is tightest in this case. Nevertheless, in experiment #1, the upper bound on the generalisation error is 30.5%, a factor of 50 above our estimate on the test set. The ratio is even worse for experiment #2.

The reader may wonder why the generalisation errors here are slightly lower than the ones reported in Table 4. This should be due to the fact that in Section 4.2, we evaluated the Bayes classifier based on the hyperparameter values which have been optimised for the Gibbs variant, while here we performed model selection for the Bayes classifier explicitly.

4.3.1 Comparison with Compression Bound for Support Vector Classifiers

We can also compare our main result for sparse GP Gibbs classifiers with state-of-the-art bounds for the popular *support vector machine (SVM)*. This kernel machine is non-probabilistic, due to its ε-insensitive loss which cannot be seen as the negative log of a proper noise distribution (see Seeger, 2000). A trained SVM discriminant function is a kernel expansion much like the discriminant function of a GP Bayes classifier (the predictive mean). The special form of the loss function encourages sparse expansions on tasks which are not very noisy. The training patterns corresponding to non-zero dual expansion coefficients are referred to as *support vectors*. However, sparseness is not a directly controllable parameter, furthermore it is not an explicit algorithmic goal of the SVM algorithm to end up with a maximally sparse expansion. The aim is rather to maximise the "soft" minimal empirical margin which is, loosely speaking, the minimal empirical margin (i.e. the distance of the discriminating hyperplane to the closest datapoints, as measured in the feature space norm) after removing some outlier training points (we are penalised for the margin violations of these outliers). This particular statistic of the empirical margin distribution is inspired by some PAC generalisation error bounds (e.g., Shawe-Taylor et al., 1998, Herbrich and Graepel, 2001), but in our opinion this link is rather weak for "non-near-asymptotic" situations. In practice, modern algorithms for SV classification such as *Sequential Minimal Optimisation (SMO)* proposed by Platt (1998) can often tackle problems with rather large sample sizes n in much less than $O(n^3)$ (average case), by concentrating optimisation efforts on a small active set. For more details on SVM see (Vapnik, 1998, Burges, 1998, Cristianini and Shawe-Taylor, 2000, Schölkopf and Smola, 2002, Herbrich, 2001). Due to the setup of SVM training as a constrained optimisation problem, it is possible to show that the algorithm is a k-compression scheme, where k is the number of support vectors. Thus, if we observe a small ratio k/n, the PAC compression bound will render a nontrivial guarantee on the generalisation error. It is important to observe that in case of SV classification, we always have $\mathrm{emp}^{\backslash k}(S) = 0$, i.e. the trained discriminant does not make any errors on the points which are not support vectors, and that this zero-error case is maximally favourable to the PAC compression bound. Experimental results for SV classifiers and the PAC compression bound can be found in Table 6. Again, we employed the same dataset splits as used in Section 4.2.

#	n	emp	gen	upper	num-sv
1	5000	0.0016	0.0048	0.2511	370.8
		(± 9.49e-4)	(±0.0012)		(±10.71)
2	9000	0.0021	0.0036	0.213	529
		(± 7.67e-4)	(±0.0012)		(±29.12)

Table 6: Experimental results for PAC compression bound with SV classifiers. n: task training set size. emp: empirical error (on full training set); gen: error on test set. upper: upper bound on generalisation error given by PAC compression bound. Figures are mean and width of 95% t-test confidence interval.

In both experiments, a higher degree of sparsity is attained than the one chosen in the experiments above for sparse GPC (as mentioned above, we did not try to optimise this degree in the sparse GPC case), leading to somewhat better values for the PAC compression bound. However, the values of 25% (experiment #1) and 21% (experiment #2) are still by factors > 50 above the estimates computed on the test set. The compression bound applies to SVM, but is certainly not specifically tailored for this algorithm, since it does not even depend on the empirical margin distribution. The margin bound of Shawe-Taylor et al. (1998), commonly used to justify data-dependent structural risk minimisation for SVM, becomes nontrivial (i.e. smaller than 1) only for $n > 34816$ (see Herbrich, 2001, Remark 4.33). The algorithmic stability bound of Bousquet and Elisseeff (2002) does not work well for support vector classification either. In fact, the gap bound value converges to zero at some rate $r(n)$ (for $n \to \infty$) only if the variance parameter[14] C goes to zero at the same rate $r(n)$. If $r(n) = O(1/n)$ or $r(n) = O(\log n/n)$, this would correspond to severe over-smoothing. Herbrich and Graepel (2001) use some older PAC-Bayesian theorems of McAllester (1999b) in order to prove a bound which depends on the minimal normalised empirical margin (SVC maximises this margin if the input points are normalised in the feature space). This theorem applies to hard-margin SVC only and becomes nontrivial once the minimal normalised (hard) margin[15] is > 0.91, given that the feature space has dimension $> n$. Hard-margin SVMs tend to overfit on noisy real-world data, with small normalised margins at least on some points, and in practice the soft-margin variant is typically preferred. In a separate experiment using the same setup and dataset splits as in #1 of this section (i.e. training sample size $n = 5000$), but training hard margin SVMs without bias parameter, we obtained generalisation error estimates on the test set of gen = 0.0056 ($\pm$ 3.904e-5), minimum normalised margins of minmarg = 0.0242 ($\pm$ 2.813e-5) and generalisation upper bound values of upper = 16.28 ($\pm$ 4.7e-3) using the theorem of Herbrich and Graepel (2001). These results back the simple observation that the minimum normalised (hard) margin is not suitable as a PAC gap bound statistic and should be replaced by one which is more robust against noise, such as soft margin, sparsity degree or combinations thereof (see Herbrich and Williamson, 2002, for some recent ideas). All in all, we were not able to find any proposed SVC-specific bound which would be tighter on this task than the PAC compression bound used above.

4.4 Using the Bounds for Model Selection

Can our results be used for model selection? In our opinion, this issue has to be approached with care. It seems rather obvious that a generalisation error bound should not be used for model selection on a real-world task if it is very far above reasonable estimates of the generalisation error on this task. When proving a PAC bound, the only link between the

14. In the SVM literature, it is common practice to separate C from the covariance kernel and write it in front of the sum over the slack variables. The parameter λ in (Bousquet and Elisseeff, 2002) is $\lambda = 2/(Cn)$, and their gap bound behaves as $1/(n\lambda)$ as $n \to \infty$.

15. If we view SVC as a linear method in a feature space induced by the covariance kernel, the minimal normalised margin is the arc cosine of the maximal angle between the normal vector of the separating plane and any of the input points mapped into feature space. A minimal normalised margin close to 1 means that *all* mapped input points lie within a double cone of narrow angle around the line given by the normal vector. For noisy data, such a situation is arguably quite unlikely to happen.

final upper bound value and the generalisation error itself that needs to be shown, is that the former dominates the latter on all but a δ-fraction of samples. Even if such a far-off bound follows the curve of a good generalisation error estimate on *some* task, there is usually no guarantee that it will do so on another one. If we minimise the upper bound value anyway for model selection, we step out of the security potentially given by the weakness of the PAC assumptions, thus could just as well use any other model selection technique such as Bayesian evidence maximisation or cross-validation.

Even though our main result offers highly non-trivial generalisation error guarantees for the real-world task described in this section, they still lie by a factor > 3 above the estimates on the test set. In our opinion, PAC generalisation error bounds *in practice* on samples of moderate size should rather be seen as secondary "safety nets" alongside a (typically) more accurate model selection criterion, such as the Bayesian evidence or a cross-validation score. To this end, the bound of course has to be tight enough to render a useful value for the sample size at hand.

In spite of these comments, we follow the usual conventions and present results of an experiment trying to assess the model selection qualities of our bound. As in Section 4.2, we use sparse greedy GPC within the setup described at the beginning of this section. The experiment consists of $L = 6$ iterations. We fixed the variance parameter C to 120 (this is sensible, given the results in Table 4) and chose a range of values for w (from 2.4 to 13, in steps of 0.2). In each iteration, we draw a training sample S of size $n = 5000$. For each configuration (C, w), w from the range, we train the method on S (using $k = 500$ and $k_{\text{rand}} = 3$), test it on the test set and also evaluate the upper bound on the expected generalisation error given by Theorem 2. It does not make sense to average the results over the L trials, so we present them all in Figure 2. In order to facilitate shape comparisons, we translate the upper bound values towards the expected test errors, by subtracting a constant (determined as 95% of the average distance between points on the two curves). In each graph, the scale printed on the left hand side is for the expected test error, the scale on the right hand side for the upper bound value. Note that the constant by which the upper bound curve is translated, as well as the curve value scales are different for each of the plots. In Figure 3, we plot expected test errors on the horizontal axis against upper bound values on the vertical axis. Note that in this type of plot, a mostly monotonically increasing relationship is what we would ideally expect. The dotted curves are lines $x + b$ with slope 1, where b is fitted to the corresponding solid curves using linear regression. The ordering of the six subplots is consistent with the ordering in Figure 2.

We see that in this particular experiment, the shape of the upper bound curve follows the shape of the expected test error rather closely, so that model selection based on minimising the upper bound value might have worked in this case. However, note that the constants we have to subtract from the upper bound curves in order to bring them close to the expected generalisation error estimates for visual inspection, are an order of magnitude larger than the range of variation of the individual curves shown in the plots. Also note that our rigid choice $k = n/10$ may be a reason behind the fairly rough behaviour of the expected test error curve. Some of the upward kinks may be due to stopping after inclusion of a bad candidate, possibly changing the posterior approximation rather drastically. Since both the expected empirical error and gap bound terms depend strongly on Q, some of these jumps

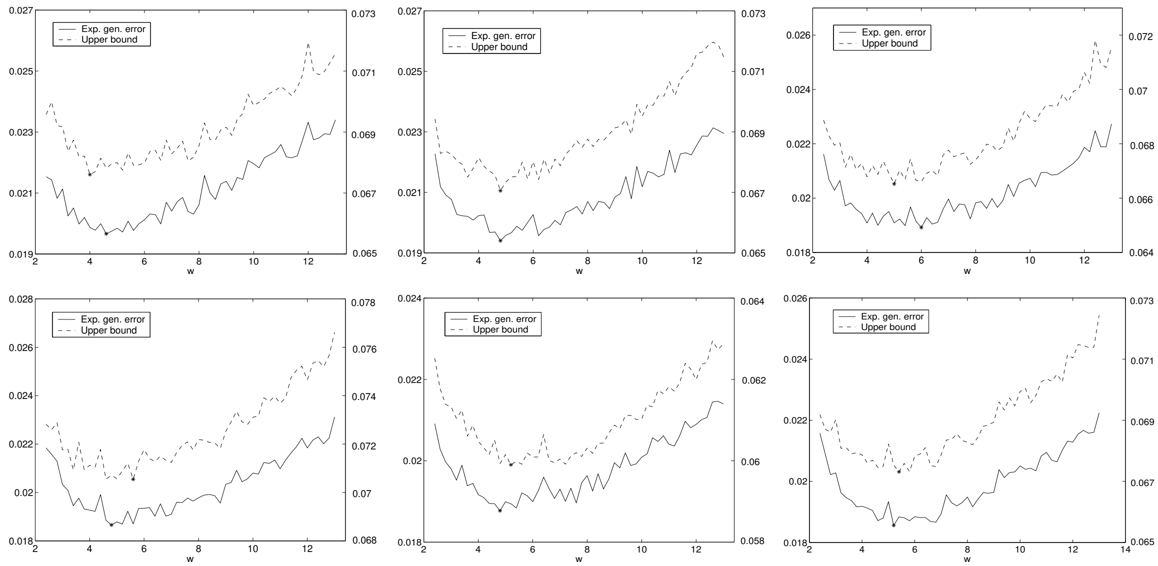

Figure 2: Comparing upper bound values with expected test errors. Solid line: expected test error (scale on left side). Dashed line: upper bound value (*translated*, scale on the right). Respective minimum points marked by an asterisk.

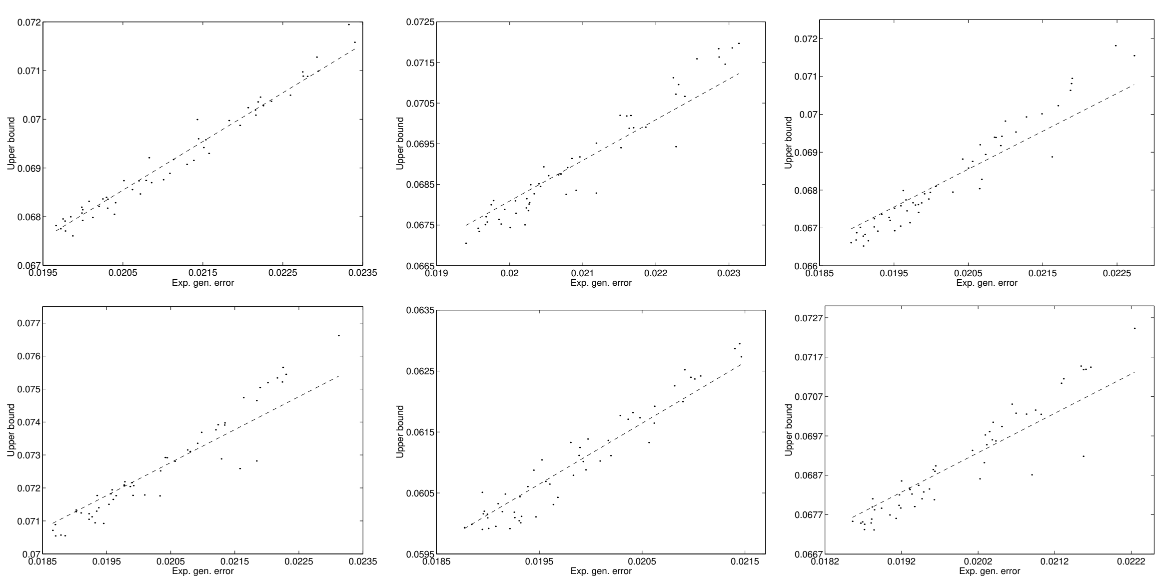

Figure 3: Comparing upper bound values (vertical axis) with expected test errors (horizontal axis). Dotted line: fitted regression line with slope 1.

induce equivalent jumps in the bound, showing just about the behaviour we would expect from a useful bound.

One might suspect that it is really only the expected empirical error which follows the expected test error closely, and that the complexity term remains fairly constant. After all, most of the points the empirical error is evaluated on are not in the active set. However, this argument ignores the fact that there *is* a dependence of the posterior on patterns outside the active set: namely, the fact that they have not been chosen for inclusion. For example, it is possible to exchange k patterns outside the active set against a set S_{new} such that another run of the algorithm will end up with the active set S_{new}. In general, a PAC bound tells us that such dependencies have to be accounted for by an additional complexity term which is of rather crude union bound type in the PAC compression bound and of a more refined type in the PAC-Bayesian theorem. Indeed, by splitting the upper bound curves above into expected empirical error and gap bound contributions, we can see that this extension is indeed necessary even if $k \ll n$. In Figure 4 we plot all these curves together in common graphs, the association of graphs with experiment iterations is the same as in Figure 2. In each graph, the two curves at the top are the upper bound (dashed) and the expected test error (solid), while the two curves at the bottom are the expected empirical error (dash-dotted) and the gap bound (dotted). The scale is correct for both the expected empirical and test error curves, while the gap and upper bound curves are translated downwards by different constants. The scale for the latter two curves is omitted.

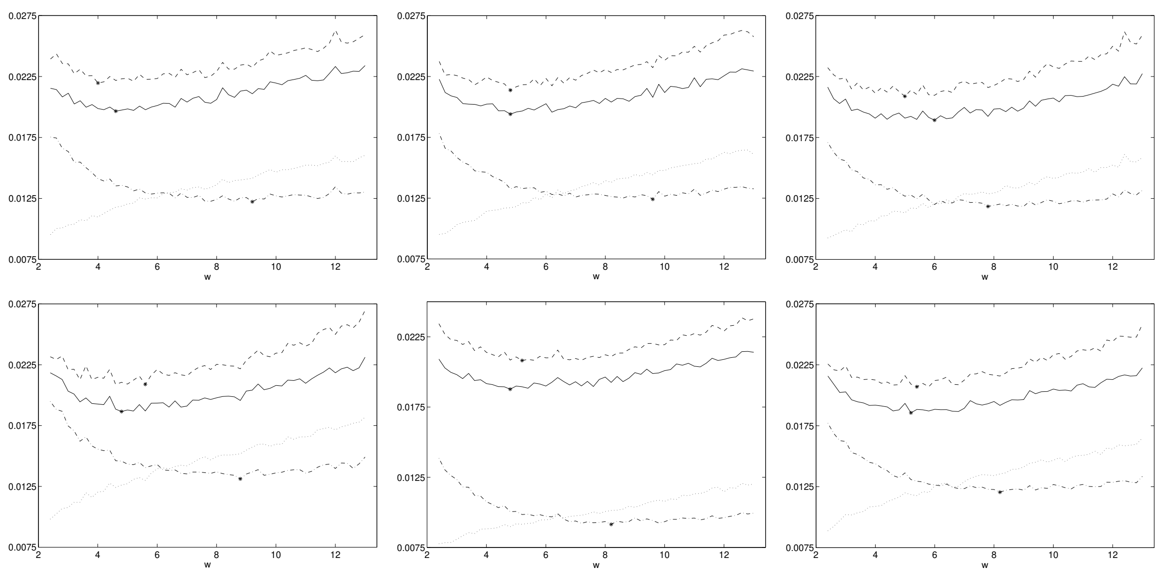

Figure 4: Comparing upper bound values with expected test errors (upper parts) and gap bound values with expected training errors (lower parts). Solid: expected test error (scale on left side). Dashed: upper bound value (*translated*). Dash-dotted: expected training error (scale on left side). Dotted: gap bound value (*translated*). Respective minimum points marked by an asterisk.

The necessity of the complexity term in our case becomes even more clear if we plot expected test errors against expected training errors in the same way as we did in Figure 3 for test errors versus upper bound values. These graphs are presented in Figure 5. We see that the linear correlation between expected training and test errors is poor. Most of the graphs in Figure 4 clearly exhibit over-fitting, in that model selection based on the expected training error would lead to preference of too large values of w, corresponding to a too narrow kernel width.

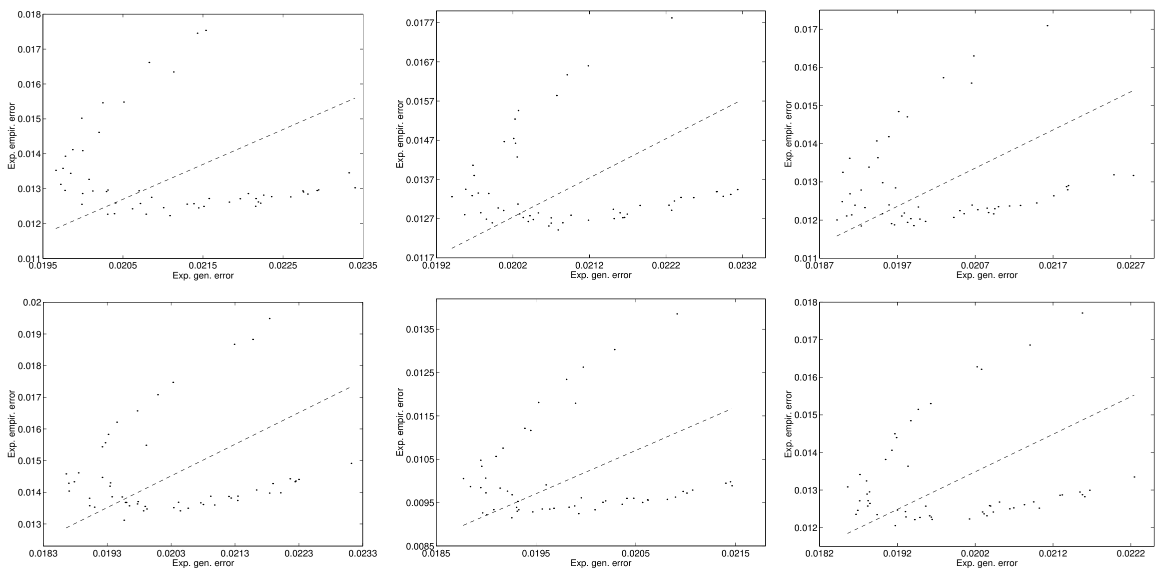

Figure 5: Comparing expected training errors (vertical axis) with expected test errors (horizontal axis). Dotted line: fitted regression line with slope 1.

5. Discussion

In this work, we have shown how to apply David McAllester's PAC-Bayesian theorem in order to obtain PAC generalisation error bounds for approximate Bayesian Gaussian process classification methods. Our main result applies to a wide class of methods, namely those that approximate the predictive process by a non-degenerate Gaussian one (e.g., Williams and Barber, 1998, Opper and Winther, 2000, Jaakkola and Haussler, 1999, Gibbs, 1997, Seeger, 2000, Csató and Opper, 2002, Lawrence, Seeger, and Herbrich, 2002). As a further contribution, we have given a simplified proof of the general PAC-Bayesian theorem.

We have derived instantiations of this result to Laplace GPC and to a class of sparse greedy GPC and tested these on a real-world task, showing that the bounds can be very tight in practically relevant situations and give more useful results than other state-of-the-art PAC bounds for kernel classifiers we considered here. One possible source of lack in tightness for many of the current data and algorithm-dependent kernel classifier bounds we know of, is that the dependence on the algorithm is actually rather weak, given only through a large margin, a certain degree of sparsity or other limited statistics, but completely independent of much of the information the algorithm has actually learned from the training sample.[16] They cannot be "configured" using prior knowledge or assumptions about the relationship between the method and the task, and therefore they cannot go very far in realizing the potential power behind the concepts of data and algorithm-dependence (see Section 1.1). These problems are directly addressed in the main result of this work (as they are in general in the PAC-Bayesian theorem). Prior knowledge can be encoded in a very general way via the choice of the covariance function, and the deviation of the sample from our assumptions is measured in a satisfactory and familiar way, by the relative entropy between approximate posterior and prior. Although, by definition every PAC bound can only depend on *some* statistics of the sample S, the particular ones we end up with in our main result are more flexible, configurable and depend more strongly on the particular algorithm than any of the ones employed in other kernel classifier bounds we know of. To conclude, although it is theoretically interesting to find general *a priori* or *a posteriori* characteristics which guarantee good generalisation performance in near-asymptotic situations over a large, even infinite set of methods, all we are really interested in in practice is to give such a guarantee for the *one* method we apply to a task, and it is not risky to conjecture that PAC bounds will have to be tailored very specifically to a given method in order to ultimately render practically useful results. As we can see from the PAC-Bayesian theorem, it is nevertheless possible to retain generality *in the theorem*.

We think that another important contribution of this work is to give an example of an effect which may be surprising at first sight: the fact that *approximate* Bayesian techniques deliberately use simplifications to overcome the intractability of exact Bayesian analysis on a model, such as decomposition or Gaussian assumptions, can often be used to simplify a PAC-Bayesian analysis of the technique *as well*. In this paper, we showed that a large class of approximations to Bayesian GPC use a Gaussian process, obtained in a simple way from the prior GP, in order to replace the true intractable posterior process, and it is *exactly* this fact that allows us to compute the corresponding relative entropy between the processes tractably as well. For a sparse GPC approximation, the computational complexity of evaluating the bound drops accordingly. Finally, relations between Gibbs and Bayes classifiers (see Section 1.3.1) can be inferred from symmetry properties of the approximate predictive distribution, while they probably do not hold for the true predictive distribution which may be skew. In short, PAC or also average-case analyses of Bayesian inference on a model might be simplified (and tightened) in many situations if instead of analysing exact intractable Bayesian inference, we focus on tractable approximations which are actually used in practice. Of course, analyses of the latter type are also of much higher interest to practitioners.

16. Actually, in most cases other, very different algorithms will attain the same (or a better) value in this statistic (or a related one) on the same sample. An example is the PAC compression bound used in Section 4.3, which actually holds uniformly for *all* methods which compress a n-sample to size k. The dominating factor in the bound comes from the fact that, in a crude union-bound argument, we have to sum over *all* of these combinatorial many possibilities (see Seeger, 2002). Another example is given in (Graepel et al., 2001), where one can infer the degree of sparsity for a kernel perceptron from the hard margin a support vector machine attains on the same sample.

5.1 Future Work

Sparse approximations of Bayesian GPC (e.g., Tipping, 2001, Williams and Seeger, 2001, Smola and Bartlett, 2001, Luo and Wahba, 1997, Tresp, 2000, Csató and Opper, 2002, Lawrence, Seeger, and Herbrich, 2002) are currently of large practical interest, and it is important to determine whether our result can be applied to them. In this paper we have shown that they apply to sparse algorithms such as proposed by Csató and Opper (2002), Lawrence, Seeger, and Herbrich (2002), Tipping (2001), and to a sparse version of the variational method of Jaakkola and Haussler (1999). In contrast to this, the Nyström method of Williams and Seeger (2001) does not employ a Gaussian process posterior approximation. A combination of the subset-of-regressors method (Wahba, 1990) with Laplace GPC uses a GP approximation to the posterior, however of a degenerate kind, leading to a trivial bound only.

In future work, we will test our result for sparse greedy GPC on more difficult tasks. To this end, the simple method we used here will have to be refined in order to increase robustness against outliers and to allow improvement of the approximation even once the desired active set size is attained. By using a motivation of the support vector machine as a limiting case of probabilistic GP classification techniques (see Opper and Winther, 2000), our Theorem 2 may imply a bound on this popular architecture as well. Furthermore, we think that the general PAC-Bayesian theorem can be rather straightforwardly applied to a host of approximate Bayesian schemes for parametric models (see also Langford and Caruana, 2002). Many of these schemes show excellent performance on real-world problems, but are not motivated by learning-theoretical analyses.

Several open problems remain. First, it would be very important to extend our results to approximate GP *Bayes* classifiers in a less crude way as has been suggested in Section 1.3.1. Typically, approximate Bayes classifiers are simpler, more efficient to evaluate, deterministic, usually perform better and are by far more frequently used in practice than the corresponding Gibbs versions. In fact, our experiments in Section 4 indicate that while the guarantees given by our main result for Gibbs kernel classifiers are tighter than guarantees for Bayes (or "Bayes-like") kernel classifiers given by state-of-the-art PAC bounds, the performance ranks we observe in practice are reversed (somewhat in test error, but especially in time requirements): this dilemma needs to be resolved, or at least understood better. As McAllester (2002) points out: *Intuitively, model averaging [the Bayes variant] should perform even better than stochastic model selection [the Gibbs variant]. But proving a PAC guarantee for model averaging superior to the PAC guarantees given here for stochastic model selection remains an open problem.*[17] Note that very recently, Meir and Zhang (2002) obtained a PAC-Bayesian margin bound for the Bayes voting classifier.

Another problem is whether the PAC-Bayesian theorem can be applied to regression estimation models, using an unbounded, convex loss (instead of the bounded, non-convex zero-one loss used in classification). This is not possible if the data distribution is completely unrestricted. The convexity of the loss implies that the risk of the Bayes classifier is less

17. The term "stochastic model selection" in (McAllester, 2002) refers to the probabilistic choice made by the Gibbs classifier on every test point. It has nothing to do with the term "model selection" used in our paper, which in the statistics literature refers to the choice of one model among a set of such, depending on criteria which are independent of the final test dataset. The term "model averaging" refers to Bayes-type classifiers.

than the expected risk of the Gibbs variant. However, under sensible restrictions on the data distribution it seems challenging to extend the proof of the PAC-Bayesian theorem to unbounded losses, even in the special case of Gaussian process regression.

The fairly close "fit" (up to a large constant!) of expected test error and the PAC-Bayesian upper bound observed in Section 4.4 indicates the usefulness of the bound in this case, yet a theoretical argument why this is the case would be very satisfying.

Finally, it would be interesting to find a way to incorporate certain widely used model selection techniques over *infinite* parameter spaces into the PAC-Bayesian framework. For example, for some of the approximations of Bayesian GPC proposed in the literature (e.g., Laplace GPC or the RVM of Tipping, 2001), it is possible to compute and optimise approximations to the *marginal likelihood*, a powerful Bayesian model selection criterion. In such cases, we can simultaneously choose good values for a large number of kernel parameters, each from within a possibly infinite set, using the training data only. In general, we will have to employ *some* sort of model selection method in order to choose free kernel parameters or other parameters of our prior. In practice, selection among a finite set of models can be incorporated by using a union bound argument, at the expense of a factor in the gap bound term which essentially replaces δ by δ/M, where M is the number of models. Such model selection techniques are, however, very limited in flexibility and scope.

A quick way around this problem is to ensure that model selection is done *independently* from the training sample S, that is we sample a separate training set to be used for model selection only. This has been done in the experiments for this paper (see Section 4). Now, from the Bayesian viewpoint, such a model selection strategy is somewhat questionable. Free parameters of the prior should be selected (if *selected* at all!) according to their posterior distribution, i.e., *conditioned* on the training data we have, and not on some independent "model selection training sample" we do not intend to use for prediction later on. Holding out part of the available training data is also wasteful.

Another idea is to simply extend the parameter $\boldsymbol{w}$ in the PAC-Bayesian theorem by the parameters of the prior and, by defining a prior distribution and computing an approximate posterior distribution over these parameters, to lift the problem one level higher in the hierarchy. However, the drawbacks of this approach for GPC techniques are severe. First of all, the relative entropy term in the bound cannot be computed analytically anymore, thus we would have to find a good analytic upper bound. Much worse, the resulting Gibbs discriminant would be very costly to evaluate, because for each evaluation we have to sample a new set of prior parameters and deal with a new covariance function for which (at least part of) a new kernel matrix has to be evaluated and a new conditioned posterior approximation has to be computed. Thus, the issue of using the PAC-Bayesian theorem together with such model selection strategies over infinite hyperparameter spaces remains without a practically satisfying solution (to our knowledge).

Acknowledgments

We thank Manfred Opper for inviting us to Aston, and for many comments which led to great simplifications of the proof of our main result, furthermore Chris Williams, Ralf Herbrich, John Langford and David McAllester for helpful discussions, and Lehel Csató and Neil Lawrence for help with the sparse GPC techniques discussed here. Chris Williams and Ralf Herbrich read several versions of the paper and made comments which led to many simplifications and improved clarity. We thank the anonymous referees for comments which led to a more readable paper. The author gratefully acknowledges support through a research studentship from *Microsoft Research Ltd.*

Appendix A. Proof of the PAC-Bayesian Theorem

In this section, we present a proof of the PAC-Bayesian Theorem 1 which is adapted to the use of classification zero-one loss employed in this work. A proof for general bounded loss functions can be found in (McAllester, 2002), making use of Hoeffding's inequality at the core, thus resulting in a less tight bound for the special case of zero-one loss. The proof we give here is considerably simpler than the one given in (McAllester, 2002).

Recall the notation introduced in Section 1. We require that there is a common dominating positive measure for the distributions P and Q over parameters. Define $p(\boldsymbol{w}, (\boldsymbol{x}, t)) = \mathrm{I}_{\{\operatorname{sgn} y(\boldsymbol{x}|\boldsymbol{w}) \neq t\}}$, furthermore let

$$p(\boldsymbol{w}) = \mathrm{E}_{(\boldsymbol{x}_*, t_*)}[p(\boldsymbol{w}, (\boldsymbol{x}_*, t_*))], \quad \hat{p}(\boldsymbol{w}) = n^{-1} \sum_i p(\boldsymbol{w}, (\boldsymbol{x}_i^S, t_i^S)),$$

where the expectation is over the unknown data distribution, and the sample is $S = \{(\boldsymbol{x}_i^S, t_i^S) \,|\, i = 1, \dots, n\}$. Let $\Delta(\boldsymbol{w}) = \mathrm{D}_{\mathrm{Ber}}[\hat{p}(\boldsymbol{w}) \,\|\, p(\boldsymbol{w})]$, where the Bernoulli relative entropy is defined in (3).

Fix $\boldsymbol{w}$, and write $\hat{p} = \hat{p}(\boldsymbol{w})$, $p = p(\boldsymbol{w})$. Then, $n\,\hat{p}$ is binomial (n, p) distributed, thus a direct computation shows

$$\begin{aligned} \mathrm{E}_S\left[e^{n \mathrm{D}_{\mathrm{Ber}}[\hat{p} \,\|\, p]}\right] &= \sum_{m=0}^{n} \binom{n}{m} \exp\left(n(\mathrm{D}_{\mathrm{Ber}}[m/n \,\|\, p] + (m/n)\log p + (1 - m/n)\log(1-p))\right) \\ &= \sum_{m=0}^{n} \binom{n}{m} e^{-nH[m/n]}, \end{aligned}$$

where $H[q] = -q \log q - (1-q)\log(1-q)$ is the entropy of a q-Bernoulli variable. But $\binom{n}{m} \le e^{nH[m/n]}$ (e.g., Cover and Thomas, 1991, Sect. 12.1), therefore

$$\mathrm{E}_S\left[e^{n \mathrm{D}_{\mathrm{Ber}}[\hat{p} \,\|\, p]}\right] \le n + 1.$$

Now, taking the average over $\boldsymbol{w} \sim P$ and using Markov's inequality, we obtain

$$\Pr_S\left\{ \mathrm{E}_{\boldsymbol{w} \sim P}\left[e^{n\,\Delta(\boldsymbol{w})}\right] > \frac{n+1}{\delta} \right\} \le \delta. \tag{21}$$

Now, fix an arbitrary sample S for which indeed

$$\mathrm{E}_{\boldsymbol{w}\sim P}\left[e^{n\,\Delta(\boldsymbol{w})}\right] \leq T, \quad T = \frac{n+1}{\delta}. \tag{22}$$

If we can show that

$$\mathrm{E}_{\boldsymbol{w}\sim Q}[n\,\Delta(\boldsymbol{w})] \leq \mathrm{D}[Q\,\|\,P] + \log \mathrm{E}_{\boldsymbol{w}\sim P}\left[e^{n\,\Delta(\boldsymbol{w})}\right], \tag{23}$$

then we have that

$$\mathrm{E}_{\boldsymbol{w}\sim Q}[\Delta(\boldsymbol{w})] \leq \frac{\mathrm{D}[Q\,\|\,P] + \log T}{n}. \tag{24}$$

In order to show (23), define the Gibbs measure

$$dP_G(\boldsymbol{w}) = \frac{e^{n\,\Delta(\boldsymbol{w})}}{\mathrm{E}_{\boldsymbol{w}\sim P}\left[e^{n\,\Delta(\boldsymbol{w})}\right]}\,dP(\boldsymbol{w}),$$

which is a probability measure relative to $P(\boldsymbol{w})$ (the definition is proper because $\exp(n\,\Delta(\boldsymbol{w}))$ is measurable w.r.t. $P(\boldsymbol{w})$). It is a well-known fact that the relative entropy between two distributions is always non-negative (e.g., Ihara, 1993, Theorem 1.4.1). Thus,

$$\begin{aligned} 0 \leq \mathrm{D}[Q\,\|\,P_G] &= \int \log\left(\frac{\mathrm{E}_{\boldsymbol{w}\sim P}\left[e^{n\,\Delta(\boldsymbol{w})}\right]}{e^{n\,\Delta(\boldsymbol{w})}}\,\frac{dQ(\boldsymbol{w})}{dP(\boldsymbol{w})}\right) dQ(\boldsymbol{w}) \\ &= \mathrm{D}[Q\,\|\,P] + \log \mathrm{E}_{\boldsymbol{w}\sim P}\left[e^{n\,\Delta(\boldsymbol{w})}\right] - \mathrm{E}_{\boldsymbol{w}\sim Q}[n\,\Delta(\boldsymbol{w})]. \end{aligned}$$

Note that what we have shown is actually a special case of a *convex duality* relation:

$$\mathrm{D}[Q\,\|\,P] = \sup_{u(\boldsymbol{w})}\left(\mathrm{E}_{\boldsymbol{w}\sim Q}\left[u(\boldsymbol{w})\right] - \log \mathrm{E}_{\boldsymbol{w}\sim P}\left[e^{u(\boldsymbol{w})}\right]\right), \tag{25}$$

meaning that the convex functions $Q \mapsto \mathrm{D}[Q\,\|\,P]$ and $u \mapsto \log \mathrm{E}_P[\exp(u(\boldsymbol{w}))]$ are in a dual relationship (i.e., Equation (25) holds as well if we interchange these functions and take the supremum over Q). (Rockafellar, 1970) is a good introduction to convex functions. Here, we only note that convex duality is a central concept in machine learning whose applications include the expectation maximisation algorithm, variational Bayesian learning and primal-dual optimisation. We can conclude the proof by noting the convexity of $\mathrm{D}_{\mathrm{Ber}}$ and using Jensen's inequality. Namely, if (24) holds for S, then

$$\mathrm{D}_{\mathrm{Ber}}\left[\mathrm{E}_{\boldsymbol{w}\sim Q}[\hat{p}(\boldsymbol{w})]\,\|\,\mathrm{E}_{\boldsymbol{w}\sim Q}[p(\boldsymbol{w})]\right] \leq \mathrm{E}_{\boldsymbol{w}\sim Q}\left[\mathrm{D}_{\mathrm{Ber}}[\hat{p}(\boldsymbol{w})\,\|\,p(\boldsymbol{w})]\right] \leq \frac{\mathrm{D}[Q\,\|\,P] + \log\frac{n+1}{\delta}}{n}. \tag{26}$$

Altogether, since $\mathrm{emp}(S,Q) = \mathrm{E}_{\boldsymbol{w}\sim Q}[\hat{p}(\boldsymbol{w})]$, $\mathrm{gen}(Q) = \mathrm{E}_{\boldsymbol{w}\sim Q}[p(\boldsymbol{w})]$, we can combine (21) and the fact that for fixed S Equation (22) implies (26), and finally invoke (5) in order to conclude that (6) must hold.

Appendix B. Summary of the Notation

We use the notation $\boldsymbol{a} = (a_i)_i = (a_1 \dots a_n)^T$ for vectors, and $\boldsymbol{A} = (a_{i,j})_{i,j}$ for matrices respectively. The superscript T denotes transposition. We use $[\cdot]$ as an operator to select parts from a matrix. Namely, for $\boldsymbol{A} \in \mathbb{R}^{m,n}$ and ordered index[18] sets $I \subset \{1, \dots, m\}$, $J \subset \{1, \dots, n\}$, $[\boldsymbol{A}]_{I,J}$ is obtained by selecting the corresponding entries (i, j), $i \in I$, $j \in J$ of $\boldsymbol{A}$ and forming a $|I| \times |J|$ sub-matrix out of them. For $I = \{k, k+1, \dots, l\}$ we write $I = k \dots l$, instead of $I = \{1, \dots, m\}$ we write $I = \cdot$, and instead of $I = \{i\}$ we write $I = i$. For example, $[\boldsymbol{A}]_{\cdot,j}$ denotes the j-th column of $\boldsymbol{A}$ and $[\boldsymbol{A}]_{i,j}$ denotes element (i, j). $\operatorname{diag} \boldsymbol{a}$ is the matrix with diagonal $\boldsymbol{a}$ and 0 elsewhere. $\operatorname{diag} \boldsymbol{A}$ is the vector containing the diagonal of $\boldsymbol{A}$. $\operatorname{tr} \boldsymbol{A}$ is the sum of the diagonal elements of $\boldsymbol{A}$. diag and tr operators have lower priority than multiplication. For example, $\operatorname{diag} \boldsymbol{A}^T \boldsymbol{b}$ is the matrix with diagonal $\boldsymbol{A}^T \boldsymbol{b}$ and 0 elsewhere. $|\boldsymbol{A}|$ denotes the determinant of $\boldsymbol{A}$. $\|\boldsymbol{a}\|$ is the Euclidean norm of the vector $\boldsymbol{a}$. With $\delta_{i,j}$, we denote the discrete Dirac delta function, i.e. $\delta_{i,j} = 1$ for $i = j$, and $\delta_{i,j} = 0$ for $i \neq j$. The identity matrix $(\delta_{i,j})_{i,j}$ is denoted by $\boldsymbol{I}$. The unit vector $(\delta_{i,j})_i$ is denoted by $\boldsymbol{\delta}_j$, the vector $(1)_i$ of all ones by $\mathbf{1}$.

If a distribution has a density, we generally use the same symbolic notation for the distribution and its density function. We use the convention of denoting a random variable and a possible value thereof with the same symbol. In general, we use $\mathrm{E}[\boldsymbol{x}]$ to denote the expectation of $\boldsymbol{x}$, and $\Pr\{A\}$ to denote the probability of an event A. By I_A, we denote the indicator function of an event A, i.e. $\mathrm{I}_A = 1$ if A is true, $\mathrm{I}_A = 0$ otherwise. $N(\boldsymbol{x}|\boldsymbol{m}, \Sigma)$ denotes the Gaussian distribution/density with mean $\boldsymbol{m}$ and covariance matrix Σ. We sometimes write $N(\boldsymbol{m}, \Sigma)$ if $\boldsymbol{x}$ is clear from the context. log denotes the logarithm to Euler's base e. The notation $f(\boldsymbol{x}) \propto g(\boldsymbol{x})$ means that $f(\boldsymbol{x}) = cg(\boldsymbol{x})$ for c constant w.r.t. $\boldsymbol{x}$. By $\operatorname{sgn} x$, we denote the sign of x, i.e. $\operatorname{sgn} x = +1$ for $x > 0$, $\operatorname{sgn} x = -1$ for $x < 0$, and $\operatorname{sgn} 0 = 0$.

References

Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

Christopher Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

Thomas Cover and Joy Thomas. *Elements of Information Theory*. Series in Telecommunications. John Wiley & Sons, 1st edition, 1991.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel Based Methods*. Cambridge University Press, 2000.

Lehel Csató and Manfred Opper. Sparse on-line Gaussian processes. *Neural Computation*, 14:641–668, 2002.

Mark N. Gibbs. *Bayesian Gaussian Processes for Regression and Classification*. PhD thesis, University of Cambridge, 1997.

18. All index sets and sets of data points are assumed to be ordered, although we use a notation known from unordered sets.

Thore Graepel, Ralf Herbrich, and Robert Williamson. From margin to sparsity. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 210–216. MIT Press, 2001.

P.J. Green and Bernhard Silverman. *Nonparametric Regression and Generalized Linear Models*. Monographs on Statistics and Probability. Chapman & Hall, 1994.

David Haussler, Michael Kearns, and Robert Schapire. Bounds on the sample complexity of Bayesian learning using information theory and the VC dimension. *Machine Learning*, 14:83–113, 1994.

David Haussler and Manfred Opper. Mutual information, metric entropy and cumulative relative entropy risk. *Annals of Statistics*, 25(6):2451–2492, 1997.

Ralf Herbrich. *Learning Kernel Classifiers*. MIT Press, 1st edition, 2001.

Ralf Herbrich and Thore Graepel. A PAC-Bayesian margin bound for linear classifiers: Why SVMs work. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 224–230. MIT Press, 2001.

Ralf Herbrich and Robert Williamson. Algorithmic luckiness. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 391–397. MIT Press, 2002.

Roger Horn and Charles Johnson. *Matrix Analysis*. Cambridge University Press, 1st edition, 1985.

Shunsuke Ihara. *Information Theory for Continuous Systems*. World Scientific, 1st edition, 1993.

Tommi Jaakkola and David Haussler. Probabilistic kernel regression models. In D. Heckerman and J. Whittaker, editors, *Workshop on Artificial Intelligence and Statistics 7*. Morgan Kaufmann, 1999.

R. E. Kaas and A. E. Raftery. Bayes factors and model uncertainty. *Journal of the American Statistical Association*, 90:773–795, 1995.

S. Kullback. *Information Theory and Statistics*. John Wiley & Sons, 1959.

John Langford and Rich Caruana. (Not) bounding the true error. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 809–816. MIT Press, 2002.

John Langford and Matthias Seeger. Bounds for averaging classifiers. Technical Report CMU-CS-01-102, Carnegie Mellon University, January 2001.

Neil D. Lawrence, Matthias Seeger, and Ralf Herbrich. Fast sparse Gaussian process methods: The informative vector machine. To appear in Neural Information Processing Systems 15, 2002.

Nick Littlestone and Manfred Warmuth. Relating data compression and learnability. Technical report, University of California, Santa Cruz, 1986.

Z. Luo and G. Wahba. Hybrid adaptive splines. *Journal of the American Statistical Association*, 92:107–116, 1997.

K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate Statistics*. Academic Press, 1st edition, 1979.

David McAllester. PAC-Bayesian model averaging. In *Conference on Computational Learning Theory 12*, pages 164–170, 1999a.

David McAllester. Some PAC-Bayesian theorems. *Machine Learning*, 37(3):355–363, 1999b.

David McAllester. PAC-Bayesian stochastic model selection. To appear in Machine Learning. See `www.autoreason.com.`, 2002.

P. McCullach and J.A. Nelder. *Generalized Linear Models*. Number 37 in Monographs on Statistics and Applied Probability. Chapman & Hall, 1st edition, 1983.

Ron Meir and Tong Zhang. Data-dependent bounds for Bayesian mixture methods. To appear in Neural Information Processing Systems 15. See `citeseer.nj.nec.com/536920.html`, 2002.

Thomas Minka. *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, Massachusetts Institute of Technology, January 2001.

Radford M. Neal. Monte Carlo implementation of Gaussian process models for Bayesian classification and regression. Technical Report 9702, Department of Statistics, University of Toronto, January 1997.

Manfred Opper and Ole Winther. Gaussian processes for classification: Mean field algorithms. *Neural Computation*, 12(11):2655–2684, 2000.

John C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*, pages 185–208. MIT Press, 1998.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 2nd edition, 1992.

R. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

Bernhard Schölkopf and Alexander Smola. *Learning with Kernels*. MIT Press, 1st edition, 2002.

Matthias Seeger. Bayesian model selection for support vector machines, Gaussian processes and other kernel classifiers. In S. Solla, T. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 603–609. MIT Press, 2000.

Matthias Seeger. PAC-Bayesian generalization error bounds for Gaussian process classification. Technical Report EDI-INF-RR-0094, Division of Informatics, University of Edinburgh, 2002. See `www.dai.ed.ac.uk/~seeger/papers.html`.

Matthias Seeger, Neil D. Lawrence, and Ralf Herbrich. Sparse Bayesian learning: The informative vector machine. Technical report, Department of Computer Science, Sheffield, UK, 2002. See `www.dcs.shef.ac.uk/~neil/papers/`.

John Shawe-Taylor, Peter L. Bartlett, Robert C. Williamson, and Martin Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Transactions on Information Theory*, 44(5):1926–1940, 1998.

Alex Smola and Peter Bartlett. Sparse greedy Gaussian process regression. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 619–625. MIT Press, 2001.

Peter Sollich. Learning curves for Gaussian processes. In M. Kearns, S. Solla, and D. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 344–350. MIT Press, 1999.

Michael Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

Volker Tresp. A Bayesian committee machine. *Neural Computation*, 12(11):2719–2741, 2000.

L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, 1st edition, 1998.

Grace Wahba. *Spline Models for Observational Data*. CBMS-NSF Regional Conference Series. SIAM Society for Industrial and Applied Mathematics, 1990.

Christopher Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001.

Christopher K. I. Williams. Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, 1997.

Christopher K.I. Williams and David Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.