Bayesian Active Learning for Sensitivity Analysis

Tobias Pfingsten[1,2]

[1] Robert Bosch GmbH, Stuttgart, Germany
[2] Max Planck Institute for Biological Cybernetics
Tobias.Pfingsten@de.bosch.com

This paper appeared in the proceedings of the ECML 2006,
LNAI 4212, pp. 354–365, Springer.

Abstract. Designs of micro electro-mechanical devices need to be robust against fluctuations in mass production. Computer experiments with tens of parameters are used to explore the behavior of the system, and to compute sensitivity measures as expectations over the input distribution. Monte Carlo methods are a simple approach to estimate these integrals, but they are infeasible when the models are computationally expensive. Using a Gaussian processes prior, expensive simulation runs can be saved. This Bayesian quadrature allows for an active selection of inputs where the simulation promises to be most valuable, and the number of simulation runs can be reduced further.
We present an active learning scheme for sensitivity analysis which is rigorously derived from the corresponding Bayesian expected loss. On three fully featured, high dimensional physical models of electro-mechanical sensors, we show that the learning rate in the active learning scheme is significantly better than for passive learning.

1 Introduction

Before computational power was widely available, general purpose simulation software hardly existed and computer models were largely tailored to answer specific questions. Today, computer models are often one-to-one emulations of physical systems and describe all their relevant features. They are usually built using powerful simulation tools—which use e.g. finite element methods to model electro-mechanical properties—and do therefore not necessarily lead to a better understanding of the system. They are used in computer experiments to replace experimental specimens.

In industrial engineering these computer experiments are often used to estimate the robustness of a design with respect to unavoidable fluctuations in mass production. Especially in the production of micro electro-mechanical systems (MEMS) such variations can significantly affect the devices' functionality. Sensitivity analysis (SA) is a standard procedure in the designing process of MEMS, and computer experiments are used to determine the influence of input parameters on the resulting fluctuation in the output. A comprehensive discussion of SA is given by [1, 2].

When fluctuations are small, SA can be done using a local approximation such as linearization. However, when this assumption does not hold, the response of the software needs to be explored over the whole range of parameter settings, and the sensitivity measures are given as expectations over the input distribution. Realistic models have tens of input parameters and the function cannot be evaluated on a regular grid. Hence, Monte Carlo (MC) methods are the most common approach for computing the expectations, where random samples from the input distribution replace the grid.

The convergence rate for MC methods is independent of the function's smoothness and the input dimension. This is certainly a useful property, but if we can use prior information—e.g. when we know that the output is a smooth function of the input parameters—we can make more efficient use of the data and save valuable simulation runs. O'Hagan [3] proposes what he calls *Bayesian quadrature*, using a Gaussian process to model the output of the simulation software, e.g. for SA [4]. Previous works have shown that, compared to MC, Bayesian quadrature can significantly improve the accuracy using the same number of randomly sampled simulation runs [2, 5].

In Bayesian quadrature we are not restricted to using samples from the input distribution and we can thus evaluate the model where the output promises to be most informative. Random sampling corresponds to what is called *passive learning* in machine learning. Actively choosing promising inputs is known as *active learning*, which has been discussed as early as 1956 by Lindley [6]. However, Bayesian active learning—also called Bayesian *experimental design*—is computationally demanding, and naturally depends strongly on what is defined to be "optimal". Therefore it cannot be considered a solved problem.

In this work we present an active learning scheme for nonparametric Gaussian process regression used in Bayesian quadrature. The learning scheme is derived as a greedy approximation to the optimal Bayesian design, where model parameters are updated after each query. We minimize the average predictive variance in the region of interest, which is derived in closed form for uniform and Gaussian input distributions. We show on three fully featured simulations of micro electromechanical sensors that the active learning scheme significantly outperforms passive learning in terms of learning rate.

We outline the Bayesian approach to active learning in section 2, discussing its relation to experimental design. Based on the generic concepts we derive a sampling scheme for sensitivity analysis in section 3. We compare the performance of passive and active learning in several experiments in section 4 and discuss the results in section 5.

2 Bayesian active learning

In the following section we discuss Bayesian active learning. Section 2.1 introduces the Bayesian concept of expected utility, which provides the formal framework for experimental design. We show in 2.2 how experimental design corresponds to active learning and define the algorithm which we use for our

experiments. For sensitivity analysis, just as for other regression setups, the objective is to minimize the expected generalization error in a region of interest. We define the corresponding utility function in 2.3.

2.1 The expected utility

Active learning is a typical example for problems which can be solved using Bayesian decision theory, where the purpose of the experiments is expressed in a utility function. The utility function will usually depend on uncertain quantities, such as model parameters and the outcomes of the experiments which are to be performed. We can therefore not directly optimize the utility function and need to average over these quantities according to our prior belief. After averaging out the unknowns, the utility function is called the *Bayesian expected utility*. Berger [7, Chap. 4] gives a comprehensive discussion of Bayesian decision theory, [8] and [9] review its application to optimal experimental design.

Assume our aim is to collect N samples, where we can choose inputs $\mathbf{x} \in \mathbb{R}^D$ at which we query targets $y \in \mathbb{R}$. We collect the targets in a vector $\mathbf{y} = (y_1, y_2 \dots y_N)^T$ and the inputs in the *design matrix* $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_N)^T$. We collect both in the *dataset* $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$. To refer to the available data at time t of decision making we use the same symbol with a time index, $\mathcal{D}_t$.

Before any optimal sampling scheme can be computed, we need to specify what is to be meant by "optimal", i.e. we need to define some *utility function*

$$U(\mathbf{X}, \mathbf{y}, \boldsymbol{\theta} | \mathcal{D}_o) \; . \tag{1}$$

The utility function usually depends on the design matrix $\mathbf{X}$ which is chosen to maximize U, the unknown outcomes of the experiments $\mathbf{y}$, and the unknown model parameters $\boldsymbol{\theta}$. The objective may also depend on the model assumption and prior information $\mathcal{D}_o$. In contrast to the remaining quantities, $\mathcal{D}_o$ is fixed.

As mentioned above, we can usually quantify the utility of a design matrix only after observing the outcomes of the experiments and for given model parameters. Bayesian decision theory provides the formalism to handle these uncertain quantities: they need to be integrated out, using the prior distribution which corresponds to $\mathcal{D}_o$,

$$U(\mathbf{X}|\mathcal{D}_o) = \int \mathrm{d}\mathbf{y} \int \mathrm{d}\boldsymbol{\theta} \quad U(\mathbf{X}, \mathbf{y}, \boldsymbol{\theta}|\mathcal{D}_o) \quad \underbrace{p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}, \mathcal{D}_o)}_{\text{model}} \underbrace{p(\boldsymbol{\theta}|\mathbf{X}, \mathcal{D}_o)}_{\text{prior}} \; . \tag{2}$$

As for the utility function we use the symbol U for the *expected utility*, simply omitting those arguments over which we have averaged.

Note, that in the expected utility (2) we assume that our prior assumptions are correct: As we average over the predictive distribution of the model $p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}, \mathcal{D}_o)$ and the prior $p(\boldsymbol{\theta}|\mathbf{X}, \mathcal{D}_o)$, the loss does not account for unexpected parameter settings or measurements which cannot be explained by the model. MacKay [10] argues that this is the "Achilles' heel" of active learning. As we assume that we are completely certain about the model, an active learning scheme

might tend to choose extreme designs which automatically confirm the model. O'Hagan discusses this problem in [11], where he introduces Gaussian processes as localized linear models. GPs relax the hard assumptions of parametric models, which can lead to designs with samples only at the limits of the input domain.

2.2 Greedy scheme for active learning

Experimental design has traditionally been used to determine a complete optimal design of N samples before any experiments are performed. The main issue is to find approximately optimal designs for large N, as the exact problem is NP-hard [12]. In the machine learning community the term "active learning" has replaced "experimental design". The focus has moved from planning a whole batch of experiments to actively planning the experiments one after the other, while updating the learning algorithm after each query. Although these approaches are quite different in their goal, both are optimally solved by maximizing the expected utility in (2).

In classical experimental design the queries $\mathbf{X}$ are planned as a batch, maximizing the expected utility $U(\mathbf{X}|\mathcal{D}_o)$. If we assume that we obtain the outcomes of all experiments at once the solution is optimal. However, if the results come one-by-one, we should refine the remaining experimental schedule in each step ℓ, by considering the measured $y_1, y_2 \dots y_\ell$ in the prior belief $\mathcal{D}_\ell$ at that time. In the Bayesian formalism it is clear that this information is correctly considered by maximizing $U(\mathbf{x}_{\ell+1} \dots \mathbf{x}_N|\mathcal{D}_\ell)$ in each query.

Most active learning schemes avoid the computational burden of planning all remaining experiments by greedily planning only one step ahead, optimizing the expected utilities $U(\mathbf{x}_{\ell+1}|\mathcal{D}_\ell)$. We use this query scheme for our experiments:

Algorithm 1 Greedy active learning

Require: N_o initial samples $\mathcal{D}_{N_o}$.
1: **for** $\ell = N_o + 1$ to N **do**
2: find $\mathbf{x}_\ell \leftarrow \operatorname{argmax}_{\mathbf{x}} U(\mathbf{x}|\mathcal{D}_{\ell-1})$.
3: query target y_ℓ to obtain new dataset $\mathcal{D}_\ell \leftarrow \mathcal{D}_{\ell-1} \cup \{(\mathbf{x}_\ell, y_\ell)\}$.
4: **end for**

2.3 Predictive performance in a region of interest

The utility function (1) formalizes the goal of the experimenter and may thus vary from problem to problem. Our aim in sensitivity analysis is to explore the output of the computer code in a region of interest, which is given by an input distribution $p(\mathbf{x})$. To measure the generalization error of the model we use its predictive variance, averaged over $p(\mathbf{x})$. Integrating out the unseen targets $\mathbf{y}$, we obtain

$$U(\mathbf{X}, \boldsymbol{\theta}|\mathcal{D}_o) = \underbrace{\int \mathrm{d}\mathbf{y}\; p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}, \mathcal{D}_o)}_{\substack{\text{average over unseen} \\ \text{training targets}}} \underbrace{\int \mathrm{d}\mathbf{x}\; p(\mathbf{x})}_{\substack{\text{average over} \\ \text{region of interest}}} \underbrace{\Big[-\operatorname{var}[y|\mathbf{x}, \boldsymbol{\theta}, \mathcal{D}, \mathcal{D}_o] \Big]}_{\substack{\text{objective: (negative)} \\ \text{pred. uncertainty}}} . \quad (3)$$

MacKay [10] discusses several utility functions to measure the generalization error in a region of interest. For the linear model information-based measures lead to so-called alphabetical designs [6, 13, 8]. While (3) is usually approximated e.g. using a sum over a pool of test cases, our setup allows for an exact solution. We derive the expected utility in the following section.

3 Active learning for nonlinear sensitivity analysis

In the preceding section we have outlined the generic concept of Bayesian active learning. We adapt the concepts to the application of Bayesian quadrature for sensitivity analysis (SA), deriving the resulting optimal sampling scheme in this section: We briefly outline SA and Bayesian quadrature in 3.1 and introduce Gaussian process regression in 3.2. We show in 3.3 how the Bayesian expected utility for SA can be derived in closed form.

3.1 Bayesian quadrature for sensitivity analysis

Global SA. The simulation software itself can be seen as a deterministic mapping from the input parameters $\mathbf{x}$ to an output $f(\mathbf{x})$. In combination with a known input distribution—which resembles fluctuations in mass production—the model can be used to determine the corresponding output distribution and the influence of single parameters.

Whenever the fluctuations are not small enough to use a local approximation of f around the nominal value, we need to use a global analysis which explores the model over the complete range of $p(\mathbf{x})$. Global sensitivity measures are thus based on expectations of the type

$$I\left[f\right] = \int \mathrm{d}\mathbf{x}\; p(\mathbf{x})\; F[f(\mathbf{x})]\;, \tag{4}$$

where F is some functional of f [1]. A first step is to compute the mean and variance of the output distribution, which are the basis for most sensitivity measures:

$$\mathrm{E}_{\mathbf{x}}[f] = \int \mathrm{d}\mathbf{x}\; p(\mathbf{x})\; f(\mathbf{x}) \qquad \text{and} \quad \mathrm{var}_{\mathbf{x}}[f] = \int \mathrm{d}\mathbf{x}\; p(\mathbf{x})\; f^2(\mathbf{x}) - \mathrm{E}_{\mathbf{x}}[f]^2\;. \tag{5}$$

Classical quadrature and Monte Carlo. The integrals (4) can be evaluated using classical quadrature if the input space is low dimensional. The error of the trapezoidal rule, for example, scales as $\mathcal{O}(N^{-2/D})$ for $F[f] \in \mathcal{C}^2$. For higher dimensions Monte Carlo (MC) estimates are to be preferred. The MC approximation to the integrals (4) is the empirical mean,

$$I\left[f\right] \approx \tfrac{1}{N} \sum_{\ell=1}^{N} F[f(\mathbf{x}_\ell)]\;, \quad \text{over samples } \mathbf{x}_\ell \text{ from } p(\mathbf{x}). \tag{6}$$

MC methods are characterized by probabilistic error bounds which scale as $\mathcal{O}(N^{-1/2})$. The MC bounds are independent of the dimension D and only require $F[f]$ to be integrable [14]. Hence, MC outperforms classical quadrature for $D \geq 5$.

Bayesian quadrature. As MC hardly makes any assumption about $F[f]$, it guarantees convergence in almost all cases. However, it is clear that the convergence rate could be better if we were able to incorporate prior knowledge into the estimates. Especially in machine learning such a trade off between bias and variance is well known to lead to a drastic improvement of the learning rate. O'Hagan [15] discusses this potential improvement of MC estimates, claiming that "Monte Carlo is fundamentally unsound".

O'Hagan [3] describes what he calls "Bayesian quadrature" to improve classical quadrature using a Gaussian process (GP) prior. Rasmussen and Williams [5] propose the "Bayesian Monte Carlo" method and show that it can outperform classical MC in high dimensions. The Bayesian quadrature scheme is:

Algorithm 2 Baysian quadrature

Require: simulation runs $\mathcal{D}$, possibly from an optimal design

1: train a Gaussian process to estimate f.
2: use the posterior $p(f|\mathcal{D})$ to estimate the integral $I[f]$ (4):

$$p(I|\mathcal{D}) = \int \mathrm{d}f \; p(f|\mathcal{D}) \; \left[\int \mathrm{d}\mathbf{x} \, p(\mathbf{x}) \; F[f(\mathbf{x})] \right] \; .$$

The posterior distribution for the integral $p(I|\mathcal{D})$ includes the remaining uncertainty. For SA the integrals for mean and variance (5) can be solved analytically.

Recall the error bounds of the trapezoidal rule and the MC method: MC hardly assumes any structure in f and its error scales as $\mathcal{O}(N^{-1/2})$ according to the strong law of large numbers. In contrast, the trapezoidal rule assumes that the function is twice differentiable and uses linear interpolation. The error $\mathcal{O}(N^{-2/D})$, guaranteed by the Taylor expansion, is better than for MC in up to four dimensions, as more structure of the function is used.

The GP regression used in Bayesian quadrature can uncover the structure of functions in high dimensional spaces, and we can therefore expect to extend the favorable convergence rate to quadrature in higher dimensions. However, the improvement comes with the cost of restricting the method to functions covered by the GP prior.

3.2 Gaussian processes applied to Bayesian quadrature

GP regression. A comprehensive introduction to GPs can be found in [16]. In the following we briefly outline the basic concepts. GPs are now widely used in machine learning, however, the model is long known for interpolation in computer experiments [17, 18].

Assume we model a mapping f from the input parameters $\mathbf{x} \in \mathbb{R}^D$ to an output $f(\mathbf{x}) \in \mathbb{R}$. Gaussian processes are defined by a (parameterized) mean and covariance function

$$\mathrm{E}[f(\mathbf{x})] = \mu(\mathbf{x}) \qquad \text{and} \qquad \mathrm{cov}\left[f(\mathbf{x}), f(\mathbf{x}')\right] = k(\mathbf{x}, \mathbf{x}') \; , \tag{7}$$

which model a known main contribution (mean) and deviations, whose structure is defined by the covariance function. We set the mean function to zero for notational simplicity, as this does not make any conceptual difference.

The GPs' behavior is governed by the choice of the covariance function $k(\mathbf{x}, \mathbf{x}')$. A common choice is to assume that correlations between the function values decay exponentially, i.e.

$$k(\mathbf{x}, \mathbf{x}') = w_o^2 \exp\left\{ -\tfrac{1}{2}\left[(\mathbf{x}-\mathbf{x}')^T A^{-1}(\mathbf{x}-\mathbf{x}')\right] \right\} \tag{8}$$

with $A = \mathrm{diag}(w_1^2, \ldots, w_d^2)$. The parameters w_o and $w_1 \ldots w_D$ control the strength of the correlations and the typical length scales of the individual input dimensions. We collect the parameters in a vector $\boldsymbol{\theta} = (w_o \ldots w_D)$.

Bayes' rule is used to combine observed data with the GP prior $p(f|\boldsymbol{\theta})$. Let the observed data $\mathcal{D}$ consist of a set of N possibly noisy observations y_ℓ of function values $f(\mathbf{x}_\ell)$. We assume normal noise, i.e. $y_\ell = f(\mathbf{x}_\ell) + \epsilon_\ell$ with $\epsilon \sim \mathcal{N}(\epsilon|0, {\sigma_y}^2)$, and add the unknown variance ${\sigma_y}^2$ to the parameter vector $\boldsymbol{\theta}$. The predictive distribution at unseen inputs $\mathbf{x}^*$ is

$$p(f^*|\mathbf{x}^*, \mathcal{D}, \boldsymbol{\theta}) = \mathcal{N}\left(f^*|m(\mathbf{x}^*), v(\mathbf{x}^*)\right) \tag{9a}$$

$$\text{with mean} \quad m(\mathbf{x}^*) = \mathbf{k}(\mathbf{x}^*)^T Q^{-1}\mathbf{y} \tag{9b}$$

$$\text{and variance} \quad v(\mathbf{x}^*) = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*)^T Q^{-1}\mathbf{k}(\mathbf{x}^*)\ ,$$

where we have defined $Q = K + \mathrm{diag}[{\sigma_y}^2, \ldots, {\sigma_y}^2]$, and used the abbreviations $\mathbf{k}(\mathbf{x}^*) \in \mathbb{R}^N$ and $K \in \mathbb{R}^{N\times N}$ with $[\mathbf{k}(\mathbf{x}^*)]_\ell = k(\mathbf{x}_\ell, \mathbf{x}^*)$ and $K_{i\ell} = k(\mathbf{x}_i, \mathbf{x}_\ell)$. As described in [7, Chap. 3] we handle the hyper parameters $\boldsymbol{\theta}$ using the maximum likelihood II (ML-II) approach, which replaces the posterior distribution for the parameters by

$$p(\boldsymbol{\theta}|\mathcal{D}) \approx \delta(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) \qquad \text{with} \quad \hat{\boldsymbol{\theta}} = \operatorname*{argmax}_{\boldsymbol{\theta}} \left[p(\mathcal{D}|\boldsymbol{\theta})\right]\ . \tag{10}$$

Bayesian quadrature. Having computed the posterior process $p(f|\mathcal{D}, \hat{\boldsymbol{\theta}})$, we can estimate mean or variance of the output under $p(\mathbf{x})$ (5) using the predictive mean and variance (9b) of the GP. The integral can be reduced to integrating products of the input distribution $p(\mathbf{x})$ and the covariance function. All necessary integrals can be computed in closed form if the covariance function (8) is used and the input distribution is Gaussian, $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{x}_o, B)$, or uniform. A Gaussian input distribution can almost always be assumed for sensitivity analysis, as it describes natural fluctuations in mass production (strong law of large numbers). The uniform distribution is appropriate for plain regression setups. The derivation of the analytic expressions is given in [2, 4].

3.3 Active learning for Bayesian quadrature

In 3.1 we have argued why Bayesian quadrature uses the available data more efficiently than MC. In the following we discuss the optimal design which improves the convergence of Bayesian quadrature.

There is a large amount of work on the design of computer experiments [19, 18, 17]. Most work reports on methods of constructing space filling designs

such as the Latin Hypercube design [20] for space filling in low dimensional projections, or the *MaxiMin* and *MiniMax* criteria [21]. These designs partly correspond to special cases of Bayesian optimal designs, but they are mostly based on intuitive considerations. The problem of computing uniform designs in high dimensional spaces is well studied. However, how to learn an appropriate distance measure and how to treat the input distribution is often not clear. Space filling designs are used in *quasi Monte Carlo* methods to improve the MC bounds by minimizing the *discrepancy* [14]. However, they are still limited to (dependent) samples from $p(\mathbf{x})$.

For Bayesian sensitivity analysis we can do more than space filling, as we explicitly know the input distribution which is naturally given by the fluctuations in mass-production. Based on the expected predictive variance over $p(\mathbf{x})$ (3) we derive a Bayesian optimal design which is exact other than using the greedy scheme (algorithm 1):

$U(\mathbf{x}_\ell|\mathcal{D}_{\ell-1})$ is given as an integral over the unknown quantities y_ℓ and $\boldsymbol{\theta}$ (2) and the input distribution $p(\mathbf{x})$ (3). The average over the parameters $\boldsymbol{\theta}$ is trivial in the ML-II framework (10) where we integrate over a δ-distribution around $\hat{\boldsymbol{\theta}}$. The integral over $p(y_\ell|\mathbf{x}_\ell, \hat{\boldsymbol{\theta}}, \mathcal{D}_{\ell-1})$ collapses as the predictive variance (9b) is independent of y_ℓ. We are left with the integral

$$U(\mathbf{x}_\ell|\mathcal{D}_{\ell-1}) = \int \mathrm{d}\mathbf{x}\; p(\mathbf{x}) \left[-\operatorname{var}\left[y|\mathbf{x}, \mathcal{D}_\ell, \hat{\boldsymbol{\theta}} \right] \right] , \tag{11}$$

which can be solved analytically. For notational simplicity we compute the utility for adding a sample $\tilde{\mathbf{x}}$ to the dataset $\mathcal{D}$ and use the definitions in (9).
The change in the predictive variance is[3]

$$\operatorname{var}[y|\mathbf{x}, \mathcal{D}, (\tilde{\mathbf{x}}, \tilde{y})] - \operatorname{var}[y|\mathbf{x}, \mathcal{D}] = -\frac{\left[k(\mathbf{x}, \tilde{\mathbf{x}}) - \mathbf{k}(\mathbf{x})^T Q^{-1} \mathbf{k}(\tilde{\mathbf{x}})\right]^2}{\operatorname{var}[\tilde{y}|\tilde{\mathbf{x}}, \mathcal{D}]} , \tag{12}$$

which, through integrating over $p(\mathbf{x})$, leads to

$$\begin{aligned} U(\tilde{\mathbf{x}}|\mathcal{D}) &= \text{const} + \int \mathrm{d}\mathbf{x}\; p(\mathbf{x}) \; \frac{\left[k(\mathbf{x}, \tilde{\mathbf{x}}) - \mathbf{k}(\mathbf{x})^T Q^{-1} \mathbf{k}(\tilde{\mathbf{x}})\right]^2}{\operatorname{var}[\tilde{y}|\tilde{\mathbf{x}}, \mathcal{D}]} \\ &= \text{const} + \frac{\left[l(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}) + \left(Q^{-1}\mathbf{y}\right)^T L \left(Q^{-1}\mathbf{y}\right) - 2\left(Q^{-1}\mathbf{y}\right)^T \mathbf{l} \right]}{\operatorname{var}[\tilde{y}|\tilde{\mathbf{x}}, \mathcal{D}]} . \end{aligned} \tag{13}$$

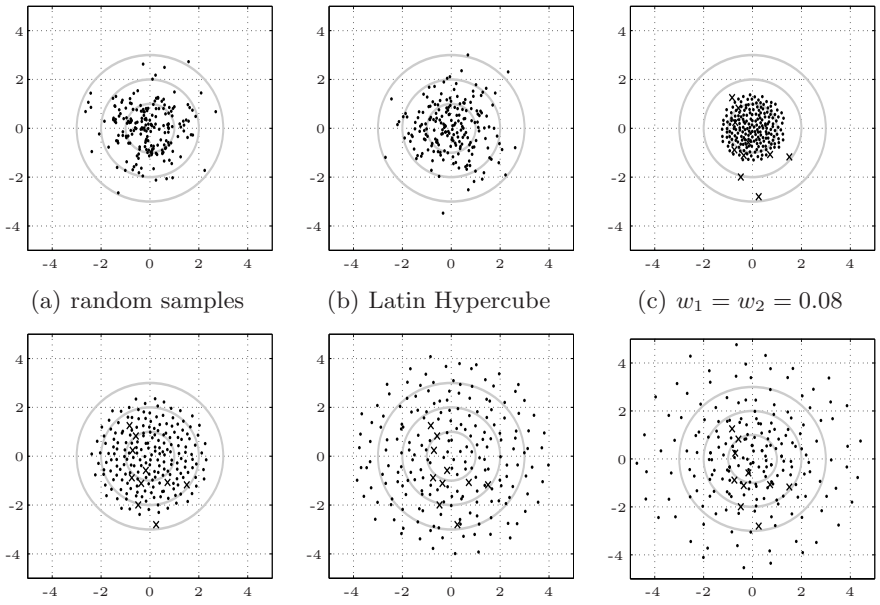
As an integral over a product of Gaussians[4] $l(\mathbf{x}', \mathbf{x}'') = \int \mathrm{d}\mathbf{x}\; p(\mathbf{x})\; k(\mathbf{x}, \mathbf{x}') k(\mathbf{x}, \mathbf{x}'')$ is easily solved analytically.

In our learning scheme we optimize (13) by using the maximum $U(\tilde{\mathbf{x}}|\mathcal{D})$ from a pool of 10 000 samples $\tilde{\mathbf{x}}$ from $p(\mathbf{x})$ and 10 000 from a Gaussian with variance $2B$, which is resampled for each draw[5].

[3] The predictive variance is given by (9b). The change for an additional sample can be derived using a rank-one update of Q^{-1}. The utility is also valid for both, noisy ($\sigma_y \neq 0$) and exact ($\sigma_y = 0$) observations y.

[4] As for k we use: $\mathbf{l}(\tilde{\mathbf{x}}) \in \mathbb{R}^N$, $L \in \mathbb{R}^{N \times N}$ with $[\mathbf{l}(\tilde{\mathbf{x}})]_\ell = l(\mathbf{x}_\ell, \tilde{\mathbf{x}})$ and $L_{i\ell} = l(\mathbf{x}_i, \mathbf{x}_\ell)$.

[5] The number of samples in the pool is somewhat arbitrary. We have made the pool large enough to obtain stable performance. To improve this brute-force optimization of



(a) random samples (b) Latin Hypercube (c) $w_1 = w_2 = 0.08$

(d) $w_1 = w_2 = 0.22$ (e) $w_1 = w_2 = 0.6$ (f) $w_1 = w_2 = 1.7$

Fig. 1. Random samples compared to optimal designs. Plot (a) shows 200 independent samples from the input distribution $p(\mathbf{x}) = \mathcal{N}(\mathbf{0}, \mathbf{1})$. Plot (b) shows a Latin Hypercube design with 200 samples. In (c–f) we have plotted optimal designs of 200 points ($\bullet$), computed using 10 initial samples ($\times$). The noise was set to a small level ($\sigma_y^2 = 10^{-5}$, $w_o = 1$). In contrast to random samples, optimal designs tend to spread the samples well apart from each other, where the length scales control the distances between the points. Latin Hypercube stratifies the design only on one dimensional projections, and uncovered areas can still be found.

Illustrative example. To visualize the difference between random samples and optimal designs, we have plotted two dimensional examples with 200 points each in figure 1. We have chosen a Gaussian input distribution with $\mathbf{x}_o = \mathbf{0}$ and $B = \mathrm{diag}(1, 1)$. Observe, in plot (a), that random samples tend to leave large areas under the input distribution uncovered, while we find some dense clusters. Naturally, most samples are found around $\mathbf{x}_o$.

Latin Hypercube sampling [20], plot (b), stratifies the design on one dimensional projections, but may show poor filling in the full space. Latin Hypercube designs sample from $p(\mathbf{x})$. Therefore they hardly provide points from low-density areas.

By considering prior measurements, the Bayesian scheme can adjust the length scales w_ℓ to reflect the variability of the function in each dimension. Plots (c–f) show the optimal designs for very short and long length scales. When the

(13) one can use a gradient based method to find the maximum, starting from several points to avoid local extrema.

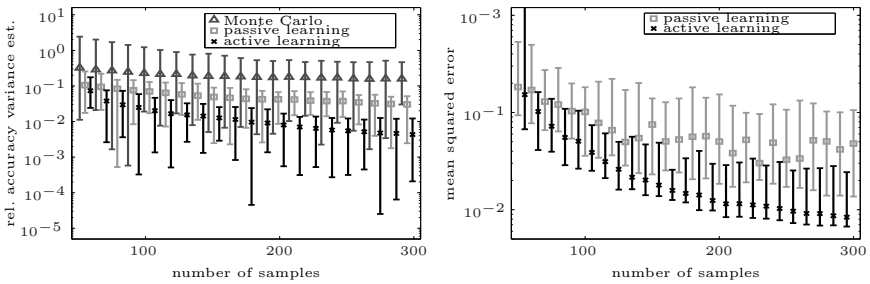


Fig. 2. Learning rates for the pressure sensor model: The estimates for the output variance using the MC method, passive and active Bayesian quadrature are plotted in the left panel. The test error for active and passive learning is shown in the right panel. The error bars indicate the median, minimum and maximum value out of 35 runs.

length scale parameter is very small ($w_1 = w_2 = 0.08$ in plot c), the correlations between function values decay rapidly and measurements need to be placed very close to each other. As the weighting factor $p(\mathbf{x})$ decreases with $|\mathbf{x}|$, the first 200 inputs are chosen close to $\mathbf{0}$. For $w_1 = w_2 = 1.7$ (f), which corresponds to a smoother function, the inputs are chosen much further apart and the input distribution is explored even where we would hardly draw a random sample from $p(\mathbf{x})$. Note that the length scales are adjusted in the ML-II scheme each time a new measurement has been observed. Hence, the sampling scheme adapts to the characteristics of the output function.

4 Experiments

When the underlying model is correct we can be certain to improve the learning rate in the Bayesian active learning scheme. It is not clear, however, how much improvement the scheme gives in real applications. The Bayesian sensitivity analysis, as presented above, is used for the design analysis of novel micro electromechanical sensors at Robert Bosch GmbH. We have tested the active learning scheme on three fully featured models of different devices, which are based on FEM simulations.

We have analyzed the model of a pressure sensor with 28 fluctuating parameters, of an accelerometer with 29 parameters, and a yaw rate sensor with 15 parameters. In all cases we have initialized the active scheme with $N_o = 20$ random samples from $p(\mathbf{x})$. To test the generalization error we have used an independent test set of 22 950, 30 000 and 50 000 samples from $p(\mathbf{x})$.

The learning curves for the pressure sensor model are shown in figure 2. The plot on the left hand side compares the accuracy of the variance estimate using the simple MC method and the Bayesian quadrature with random and actively chosen samples. On 300 samples the Bayesian quadrature is by an order of magnitude more accurate than the MC method, and by using active learning we gain another factor of five. The plot to the right shows the mean squared error on the test set, which reflects this improvement.

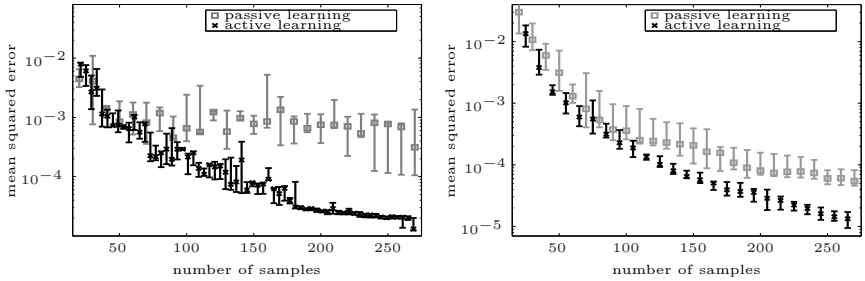


Fig. 3. Learning curves for the model of the accelerometer (left) and the yaw rate sensor (right). The markers indicate the median of 6 (left) and 3 (right) runs, the error bars cover the interval from the minimal to the maximal value.

For the accelerometer and the yaw rate sensor we show the learning curves in figure 3. As in the other example, the active scheme clearly outperforms passive learning. At 270 samples we gain roughly a factor five in accuracy. Note, that the performance for random sampling scatters much stronger than that of the active scheme, as the latter is only partly randomized.

5 Discussion

Monte Carlo estimates are commonly used to explore the global behavior of computer models for sensitivity analysis. They have the advantage that they are simple to implement and that they do not make strong assumptions about the structure of the output. However, MC may not be feasible when the function is computationally too complex to be evaluated at a great number of parameter settings.

In industrial engineering computer experiments often model the behavior of complete physical systems, and they are used to analyze the robustness of a design with respect to fluctuations in mass production. For many models a sensitivity analysis is not feasible using the MC approach. Bayesian quadrature can resolve the problem by using the available data more efficiently.

In SA we are given—in contrast to most benchmark problems in machine learning—the region of interest and simulation software which can be called at any input. We can therefore use an active learning scheme which calls the software where the evaluation promises most informative. In contrast to previous work, which mainly uses space filling designs, our approach directly optimizes the Bayesian expected utility and updates the model parameters in each step. As the input distribution in SA is Gaussian, we can compute the expected utility analytically.

To quantify the benefit of the active learning scheme we have used three high dimensional, fully featured models from industrial engineering, which resemble micro electro-mechanical sensors. The models have up to 29 uncertain inputs, where the input distributions reflect fluctuations from mass production.

Bayesian quadrature proves much more efficient than the simple MC method. Compared to passive Bayesian quadrature with random samples from $p(\mathbf{x})$, the active learning scheme leads to a significant improvement of the generalization error. By using a uniform input distribution, the learning scheme can be applied to standard regression setups.

References

1. Saltelli, A., Chan, K., Scott, E.M.: Sensitivity Analysis. Wiley (2000)
2. Pfingsten, T., Herrmann, D.J., Rasmussen, C.E.: Model-based design analysis and optimization. IEEE Trans. on Semiconductor Manufacturing (accepted) (2006)
3. O'Hagan, A.: Bayes-Hermite Quadrature. Journal of Statistical Planning and Inference **29**(3) (1991) 245–260
4. Oakley, J.E., O'Hagan, A.: Probabilistic sensitivity analysis of complex models: a Bayesian approach. Journal of the Royal Statistical Society, Series B **66**(3) (2004) 751–769
5. Rasmussen, C.E., Ghahramani, Z.: Bayesian Monte Carlo. In: NIPS 15. (2003)
6. Lindley, D.V.: On the measure of information provided by an experiment. Ann. math. Statist. **27** (1956) 986–1005
7. Berger, J.O.: Statistical Decision Theory and Bayesian Analysis. Springer New York (1985)
8. Chaloner, K., Verdinelli, I.: Bayesian experimental design: A review. Statistical Science **10**(3) (1995) 273–304
9. Lindley, D.: Bayesian Statistics — A Review. SIAM (1972)
10. Mackay, D.: Information-based objective functions for active data selection. Neural Computation **4**(4) (1992) 589–603
11. O'Hagan, A.: Curve Fitting and Optimal Design for Prediction. J.R. Statist. Soc. B **40**(1) (1978) 1–42
12. Ko, C.W., Lee, J., Queyranne, M.: An exact algorithm for maximum entropy sampling. Operations Research **43**(4) (1995) 684–691
13. Lindley, D.: The choice of variables in multiple regression. Journal of the Royal Statistical Society B **30**(1) (1968) 31–66
14. Niederreiter, H.: Random number generation and quasi-Monte Carlo methods. SIAM (1992)
15. O'Hagan, A.: Monte Carlo is Fundamentally Unsound. The Statistician **36**(2/3) (1987) 247–249
16. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. MIT Press (2006)
17. Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P.: Design and analysis of computer experiments. Statistical Science **4**(4) (1989) 409–423
18. Welch, W.J., Buck, R.J., Sacks, J.S., Wynn, H.P., Mitchell, T.J., Morris, M.D.: Screening, prediction, and computer experiments. Technometrics **34**(1) (1992) 15–25
19. Santner, T.J., Williams, B.J., Notz, W.I.: The Design and Analysis of Computer Experiments. Springer New York (2003)
20. Mackay, M.D., Beckmann, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics **21**(2) (1979) 239–245
21. Johnson, M.E., Ylvisaker, D., Moore, L.: Minimax and maximin distance designs. J. of Statistical Planning and Inference **26** (1990) 131–148