LOMONOSOV MOSCOW STATE UNIVERSITY
FACULTY OF COMPUTATIONAL MATHEMATICS AND CYBERNETICS
CHAIR OF MATHEMATICAL METHODS OF FORECASTING

IZMAILOV PAVEL

Gaussian Processes for Machine Learning

COURSE WORK

Scientific advisors:
D. P. Vetrov
D. A. Kropotov

Moscow, 2016

Contents

Abstract

Gaussian processes provide an elegant and effective approach to learning in kernel machines. This approach leads to a highly interpretable model and allows using the bayesian framework for model adaptation and incorporating the prior knowledge about the problem. Unfortunately, the standard methods for GP-regression and GP-classification scale as $\mathcal{O}(n^3)$, where n is the size of the dataset, which makes them inapplicable for big data problems. In this work we describe two modern methods for GP-regression and one for GP-classification, which attempt to solve this problem, and provide their experimental comparison.

1 Introduction

Gaussian processes provide a prior over functions and allow finding complex regularities in the data. In this section we describe, what Gaussian process is and how to fit it to the data in regression and classification settings.

1.1 Gaussian process definition

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

We will only consider processes, that take place in a finite dimensional real space $\mathbb{R}^d$. In this case, f is a Gaussian process, if for any k, for any $t_1, t_2, \ldots, t_k \in \mathbb{R}^d$ the joint distribution

$$(f(t_1), f(t_2), \ldots, f(t_k))^T \sim \mathcal{N}(m_t, K_t)$$

for some m_t and K_t.

The mean m_t of this distribution is defined by the mean function $m : \mathbb{R}^d \to \mathbb{R}$:

$$m_t = (m(t_1), m(t_2), \ldots, m(t_k))^T.$$

Similarly, the covariance matrix K_t is defined by the covariance function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$:

$$K_t = \begin{pmatrix} k(t_1,t_1) & k(t_1,t_2) & \ldots & k(t_1,t_n) \\ k(t_2,t_1) & k(t_2,t_2) & \ldots & k(t_2,t_n) \\ \ldots & \ldots & \ldots & \ldots \\ k(t_n,t_1) & k(t_n,t_2) & \ldots & k(t_n,t_n) \end{pmatrix}.$$

It's straightforward then, that a Gaussian process is completely defined by its mean and covariance functions. We will use the following notation.

$$f \sim \mathcal{GP}(m(\cdot), k(\cdot,\cdot))$$

means that f is a Gaussian process with mean function m and covariance function k. While the mean function can be an arbitrary real-valued function, the covariance function has to be a kernel, so that the covariance matrices it implies are symmetric and positive definite.

Fig. 1 shows an example of a one-dimensional Gaussian process. The dark blue line is the mean of the process, and the light blue region is the 3σ-region, drawn at each point.

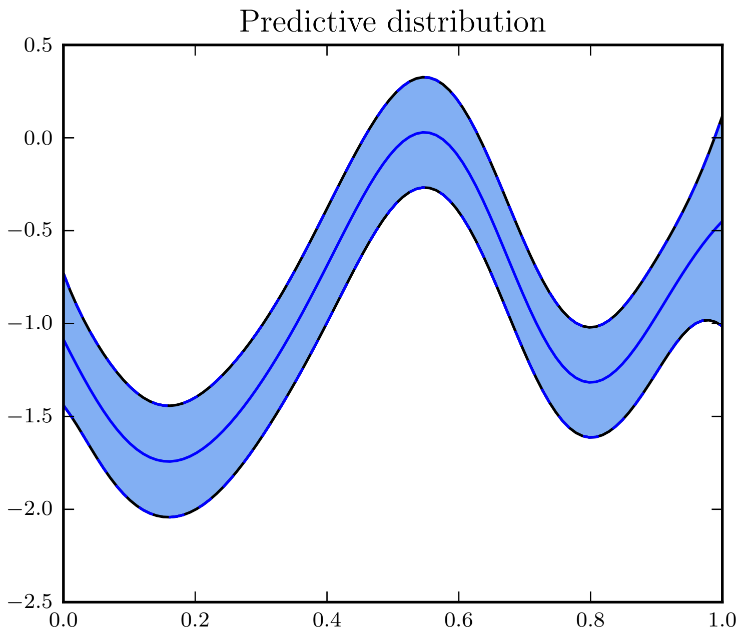


Fig. 1: One-dimensional Gaussian processes

1.2 Gaussian process regression

Consider the regression problem. We have a dataset $\{(x_i, f_i) | i = 1, \dots, n\}$, which is considered to be generated from an unknown Gaussian process $f \sim \mathcal{GP}(m(\cdot), k(\cdot, \cdot))$, let $x \in \mathbb{R}^d$. We will denote the matrix comprised of points $x_1, \dots, x_n$ by $X \in \mathbb{R}^{n \times d}$ and the vector of corresponding target values $f_1, ..., f_n$ by $f \in \mathbb{R}^n$. We want to predict the values $f_* \in \mathbb{R}^l$ of the unknown process at a set of other l points $X_* \in \mathbb{R}^{l \times d}$.

We will consider the case, when we can't directly observe the values f of the process at points X. Instead, we will use their noisy versions y, which are distributed as

$$y \sim \mathcal{N}(f, \nu^2 I),$$

for some noise variance $\nu \in \mathbb{R}$. If $\nu = 0$ we obtain the noise-free case $y = f$.

We now put a zero-mean Gaussian process prior on our model

$$f \sim \mathcal{GP}(0, k(\cdot, \cdot)).$$

This prior might seem limiting, however a zero-mean prior does not imply a zero-mean predictive distribution.

The probabilistic model for this setting is given by

$$p(y, f | X) = p(y | f, X) p(f | X).$$

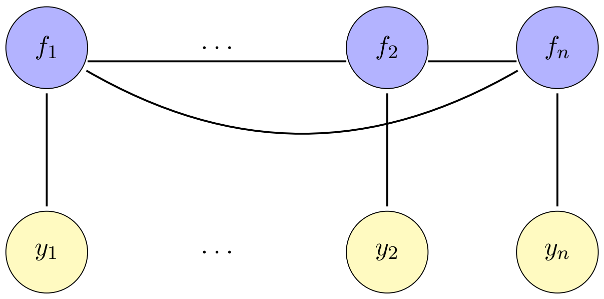


Fig. 2: The graphical model for Gaussian process regression and classification

In the following text we will omit X in order to make the formulas simpler. We then rewrite the model in the following form

$$p(y, f) = p(y|f)p(f) = p(f)\prod_{i=1}^{n} p(y_i|f_i). \tag{1}$$

The graphical model for this setting is provided in fig. 2.

We will use the following notaion. We will denote the matrix, comprised of pairwise values, of covariance functions on two sets of points $A = (a_1, \dots, a_n)^T \in \mathbb{R}^{n\times d}$ and $B = (b_1, \dots, b_m)^T \in \mathbb{R}^{m\times d}$ by

$$K(A, B) = \begin{pmatrix} k(a_1, b_1) & k(a_1, b_2) & \dots & k(a_1, b_m) \\ k(a_2, b_1) & k(a_2, b_2) & \dots & k(a_2, b_m) \\ \dots & \dots & \dots & \dots \\ k(a_n, b_1) & k(a_n, b_2) & \dots & k(a_n, b_m) \end{pmatrix} \in \mathbb{R}^{n\times m}.$$

Then, by definition of a Gaussian process the joint distribution of f and f_* is given by

$$\begin{bmatrix} f \\ f_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X, X) & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right).$$

As y is obtained by adding a normally distributed noise to f, the joint distribution

$$\begin{bmatrix} y \\ f_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X, X) + \nu^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right).$$

Conditioning this distribution, we obtain the predictive

$$f_*|y \sim \mathcal{N}(\hat{m}, \hat{K}),$$

where

$$\mathbb{E}[f_*|y] = \hat{m} = K(X_*, X)(K(X, X) + \nu^2 I)^{-1} y,$$
$$\operatorname{cov}(f_*|y) = \hat{K} = K(X_*, X_*) - K(X_*, X)(K(X, X) + \nu^2 I)^{-1} K(X, X_*).$$

Thus, the complexity of obtaining the predictive distribution is determined by the complexity of inverting the n by n matrix $K(X, X) + \nu^2 I$ and thus scales as $\mathcal{O}(n^3)$.

In fig. 3 you can see examples of one-dimensional Gaussian processes, reconstructed from data. The data points are shown by black '+' signs.

For more detailed description of Gaussian process regression see [1].

1.3 Gaussian process classification

Now we will apply the Gaussian processes to the binary classification problem, which can be described as follows. We have a dataset $\{(x_i, y_i) | i = 1, \dots, n\}$, where $x_i \in \mathbb{R}^d$, $y_i \in \{-1, 1\}$. We want to predict whether or not a new data point x_* belongs to the positive class, given its coordinates.

We will consider the following model. We will introduce a latent function $f : \mathbb{R}^d \to \mathbb{R}$ and put a zero-mean GP prior over it.

$$f \sim \mathcal{GP}(0, k(\cdot, \cdot)).$$

We will then consider the probability of the object x_* belonging to positive class to be equal to $\sigma(f(x_*))$ for the chosen sigmoid function σ.

$$p(y_* = +1 | x_*) = \sigma(f(x_*)).$$

Note, that the graphical for this model is exactly the same, as for the regression problem and is given in fig. 2.

We will use the logistic function $\sigma(z) = (1 + \exp(-z))^{-1}$, however one can use other sigmoid functions as well.

Now inference can be done in two steps. First, for the new data point x_* we should find the conditional distribution of the value of the latent process f at the new data point x_*. This can be computed as follows

$$p(f_* | y) = \int p(f_* | f) p(f | y) df. \tag{2}$$

Now, the probability that x_* belongs to the positive class is obtained by marginalizing over the latent variable f_*.

$$p(y_* = +1 | y) = \int \sigma(f_*) p(f_* | y) df_*. \tag{3}$$

Unfortunantely, both integrals in (2) and (3) are intractable. Thus, we have to use integral-approximation techniques to estimate the predictive distribution.

For example, one can use Laplace approximation method, which builds a Gaussian approximation $q(f | X, y)$ to the true posterior $p(f | X, y)$. This approximation is obtained by performing the Taylor expansion of the function $\log p(f | X, y)$ around its maximum $\hat{f}$.

Substituting this Gaussian approximation back into (2) and (3), we obtain tractable integrals and can compute the predictive distribution in a closed form. The more detailed derivation of this algorithm and another algorithm, based on Expectation Propagation can be found in [1].

We will also describe another method for GP-classification below.

Computational complexity of computing the predictive distribution for this method scales as $\mathcal{O}(n^3)$.

1.4 Model adaptation

In two previous sections we described, how to fit a Gaussian process to the data in regression and classification problem. However, we only considered the Gaussian processes with fixed covariance functions. This model can be rather limiting.

Most of the popular covariance functions have a set of parameters, which we will refer to as covariance (or kernel) hyper-parameters. For example, the squared exponential covariance function

$$k_{SE}(x, x') = \sigma^2 \exp\left(-\frac{\|x - x'\|^2}{l^2}\right)$$

has two parameters — variance σ and length-scale l. An example of a more complicated popular covariance function is the Matern function, given by

$$k_{Matern}(x, x') = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}\|x - x'\|}{l}\right)^{\nu} K_{\nu}\left(\frac{\sqrt{2\nu}}{\|x - x'\|} l\right),$$

with two positive parameters ν and l. Here K_ν is a modified Bessel function.

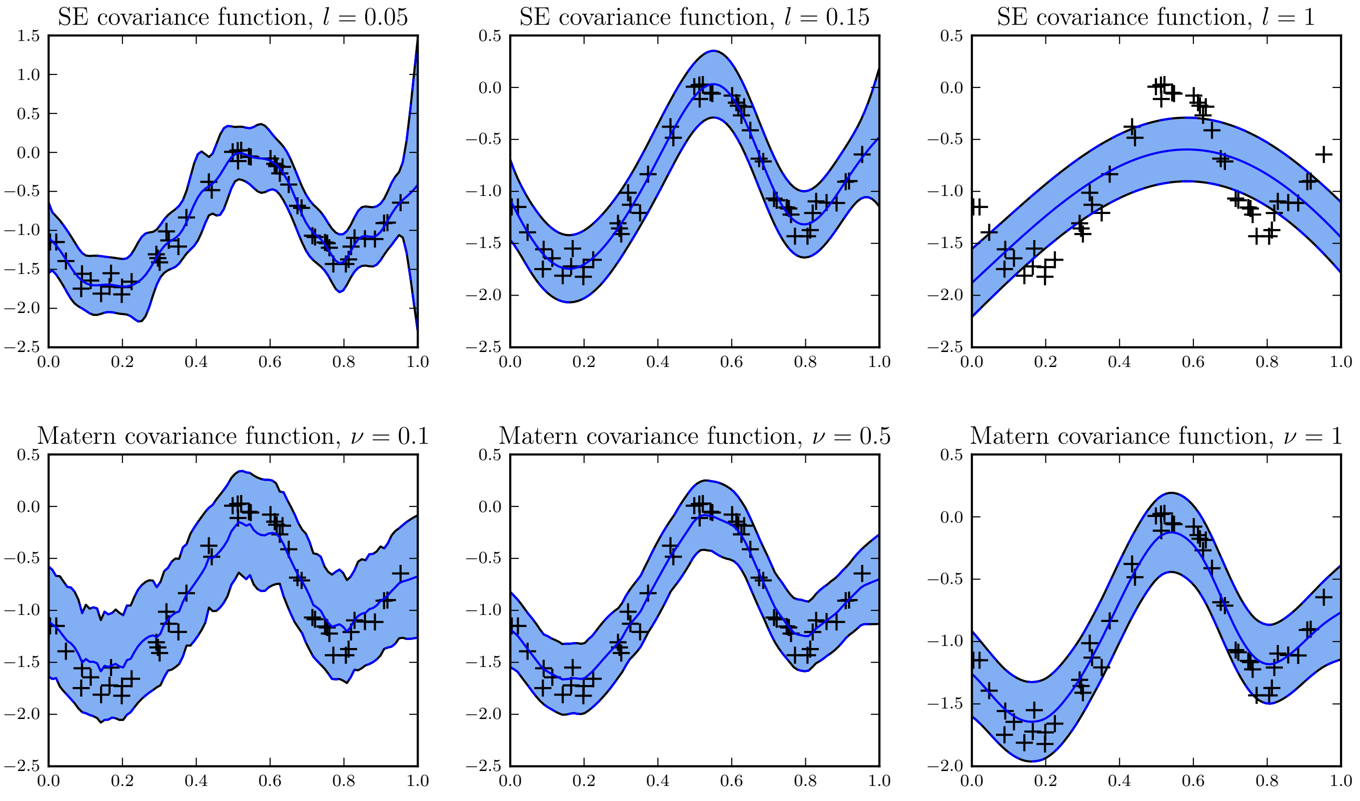


Fig. 3: Gaussian processes with squared exponential and Matern covariance functions, reconstructed from the same data for different values of hyper-parameters

In fig. 3 you can see the predictive distributions of the Gaussian-process regression for the same dataset for different values of kernel hyper-parameters of the squared exponential and Matern covariance functions. It can be seen from these plots, that in order to get a good model for the data, one should find a good set of kernel hyper-parameters.

Bayesian paradigm provides a way of tuning the kernel hyper-parameters of the GP-model through maximization of the evidence, or marginal likelihood of the model. Marginal likelihood is given by

$$p(y) = \int p(y|f)p(f)df,$$

which is the likelihood, marginalized over the latent values f of the underlying process.

For the GP-regression the marginal likelihood can be computed in closed form and is given by

$$\log p(y|X) = -\frac{1}{2}y^T(K + \nu^2 I)^{-1}y - \frac{1}{2}\log|K + \nu^2 I| - \frac{n}{2}\log 2\pi. \tag{4}$$

For the Laplace approximation method, the marginal likelihod is also available in the closed form.

Thus, for the regression problem, we should first maximize the evidence, given by (4) with respect to covariance hyper-parameters and then use the predictive distribution, derived in section 1.2, to make predictions for the new data points.

For the Laplace approximation method for the classification problem, the procedure is slightly more complicated and is described in [1].

2 Approximate methods

In the previous section we've described two methods for applying the Gaussian processes to the regression and binary classification problems. Both these methods scaled as $\mathcal{O}(n^3)$, where n is the size of the training set. The computational complexity makes these methods inapplicable to big data problems, and thus approximation methods are needed.

A number of approximate methods have been proposed in the literature. We will consider methods based on the concept of inducing inputs. These methods construct an approximation based on the values of the process at some $m < n$ points. These points are referred to as inducing points. The first methods of this kind choose the inducing points from the training set heuristically or through greedy optimization of some criterion. For a review of these methods see, for example [9]. Paper [10] provides a numerical comparison of several sparse GP methods.

We will consider the variational approach to selecting the inducing variables. In this approach the inducing inputs are not limited to belong to the training set and their positions as well as the process values at these points can be learned jointly with the values of kernel hyper-parameters. There are several methods for both regression and classification problems, based on this approach, which we will derive and compare in the following sections.

2.1 Evidence lower bound

In order to derive the lower bound, we change our probabilistic model by introducing a set of latent variables u, so that

$$p(y, f, u) = p(y|f)p(f|u)p(u) = \prod_{i=1}^{n} p(y_i|f_i)p(f|u)p(u). \tag{5}$$

The corresponding graphical model is shown in fig. 4. The values u are considered to be the values of the same Gaussian process, from which the dataset is generated, at a set of points $Z = (z_1, \dots, z_m)^T \in \mathbb{R}^{m\times d}$. Thus, the distribution $p(u)$ is given by

$$p(u) = \mathcal{N}(u|0, K(Z, Z)).$$

Points Z are called inducing points or inducing inputs. To make the formulas more clear, we will use a simpler notation for covariance matrices. We will denote $K(Z, Z)$ by K_{mm}, $K(X, Z)$ by K_{nm}, $K(Z, X)$ by K_{mn}, and $K(X, X)$ by K_{nn}.

We will refer to the introduced model as augmented model. Note that integrating (5) with respect to u we obtain the standard model (1).

As u and f are generated from a Gaussian process with zero-mean prior,

$$p(u) = \mathcal{N}(u|0, K_{mm}),$$

$$p(f|u) = \mathcal{N}(f|K_{nm}K_{mm}^{-1}u, \tilde{K}),$$

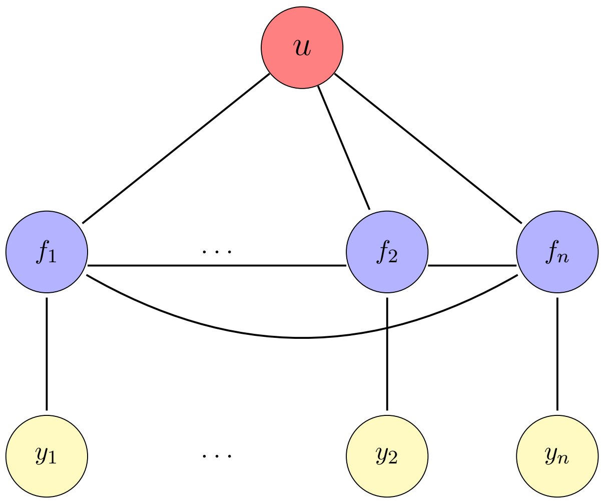


Fig. 4: The augmented model

where $\tilde{K} = K_{nn} - K_{nm}K_{mm}^{-1}K_{mn}$.

Applying the standard variational lower bound (see, for example [11]) to the augmented model, we obtain the following inequality.

$$\log p(y) \geq \mathbb{E}_{q(u,f)} \log \frac{p(y,u,f)}{q(u,f)} = \mathbb{E}_{q(u,f)} \log p(y|f) - \mathrm{KL}\left(q(u,f) \;||\; p(u,f)\right),$$

for any distribution $q(u,f)$. This inequality becomes equality for the true posterior distribution $q(u,f) = p(u,f|y)$. We will restrict $q(u,f)$ to be of the form

$$q(u,f) = p(f|u)q(u),$$

where $q(u) = \mathcal{N}(u|\mu,\Sigma)$ for some $\mu \in \mathbb{R}^m$, $\Sigma \in \mathbb{R}^{m\times m}$.

This form of the variational distribution implies a Gaussian marginal distribution

$$q(f) = \int p(f|u)q(u)du = \mathcal{N}(f|K_{nm}K_{mm}^{-1}\mu, K_{nn} + K_{nm}K_{mm}^{-1}(\Sigma - K_{mm})K_{mm}^{-1}K_{mn}).$$

As $\log p(y|f)$ depends on u only through f, the expectation $\mathbb{E}_{q(u,f)} \log p(y|f) = \mathbb{E}_{q(f)} \log p(y|f)$. Now, as the likelihood factorizes over objects, $\mathbb{E}_{q(f)} \log p(y|f) = \sum_{i=1}^{n} \mathbb{E}_{q(f_i)} \log p(y_i|f_i)$, where $q(f_i)$ is the marginal distribution of $q(f)$.

Finally,

$$\mathrm{KL}\left(q(u,f) \;||\; p(u,f)\right) = \mathrm{KL}\left(q(u)p(f|u) \;||\; p(u)p(f|u)\right) = \mathrm{KL}\left(q(u) \;||\; p(u)\right).$$

Combining everything back together, we obtain the evidence lower bound

$$\log p(y) \geq \sum_{i=1}^{n} \mathbb{E}_{q(f_i)} \log p(y_i|f_i) - \mathrm{KL}\left(q(u) \;||\; p(u)\right). \tag{6}$$

Note that the KL-divergence term in the lower bound (6) can be computed analytically, as it is a KL-divergence between two normal distributions. The expectations $\mathbb{E}_{q(f_i)} \log p(y_i|f_i)$ can be computed analytically in case of the regression problem. In case of classification we have to use integral approximating techniques in order to compute these one-dimensional integrals.

The evidence lower bound (ELBO) (6) can be maximized with respect to variational parameters μ, Σ and kernel hyper-parameters. Using the optimal distribution $q(u)$, we can perform predictions for new data point x_* as follows

$$p(f_*|y) = \int p(f_*|f,u)p(f|u,y)p(u|y)dudf \approx \int p(f_*|f,u)p(f|u)q(u)dudf = \int p(f_*|u)q(u)du.$$

This integral is tractable and

$$p(f_*|y) \approx \mathcal{N}(f|K(x_*,Z)K_{mm}^{-1}\mu, K(x_*,x_*) + K(x_*,Z)K_{mm}^{-1}(\Sigma - K_{mm})K_{mm}^{-1}K(Z,x_*)).$$

Below we will consider three different methods for optimizing the lower bound (6).

2.2 VI method

The first method we will describe was introduced in [2]. In case of the regression problem we can analytically optimize the bound (6) with respect to variational parameters. Differentiating the lower bound (6) and setting the derivatives to 0, we obtain the optimal values of variational parameters.

$$\hat{\Sigma} = \left(\frac{1}{\nu}K_{mm}^{-1}K_{mn}K_{nm}K_{mm}^{-1} + K_{mm}^{-1}\right)^{-1},$$

$$\hat{\mu} = \frac{1}{\nu}\hat{\Sigma}K_{mm}^{-1}K_{mn}y.$$

Now, substituting the optimal values of variational parmeters back into 6, we obtain a new lower bound

$$\log p(y) \geq \log \mathcal{N}(y|0, \nu^2 I + K_{nm}K_{mm}^{-1}K_{mn}) - \frac{1}{2\nu^2}\text{tr}(\tilde{K}), \tag{7}$$

where $\tilde{K} = K_{nn} - K_{nm}K_{mm}^{-1}K_{mn}$.

This lower bound and its derivatives with respect to covariance hyper-parameters can be computed in $\mathcal{O}(nm^2) + \mathcal{O}(m^3)$ operations. As m is considered to be sustantially smaller than n, this is equivalent to $\mathcal{O}(nm^2)$. This complexity makes the method applicable to moderate and even big problems.

Note, that we can also optimize the bound with respect to the positions Z of the inducing inputs.

We will call this method `vi` (variational inference) as opposed to `svi` (stochastic variational inference) methods, described in the latter sections.

2.3 SVI method

In this subsection we describe a method for maximizing the lower bound (6) in case of the GP-regression problem, which was proposed in [3]. While the method described in the previous section is much faster then the full GP-regression, it's complexity is still rather big. We could try to reduce the time consumption of optimizing the lower bound by using stochastic optimization methods. However, the function in the right-hand side of (7) does not have a form of sum over objects, and thus it's not clear, how to apply the stochastic methods.

However, the original bound from (6) does have a form of sum over objects, and we can thus apply stochastic methods to it. In the regression case the expectations in the bound (6) are tractable. In this case, we can rewrite the bound as

$$\log p(y) \geq \sum_{i=1}^{n} \left(\log \mathcal{N}(y_i | k_i^T K_{mm}^{-1} \mu, \nu^2) - \frac{1}{2\nu^2} \tilde{K}_{ii} - \frac{1}{2} \mathrm{tr}(\Sigma \Lambda_i) \right) - \\ - \frac{1}{2} \left(\log \frac{|K_{mm}|}{|\Sigma|} - m + \mathrm{tr}(K_{mm}^{-1} \Sigma) + \mu^T K_{mm}^{-1} \mu \right), \quad (8)$$

where $\Lambda_i = \frac{1}{\nu^2} K_{mm}^{-1} k_i k_i^T K_{mm}^{-1}$, and $k_i = K(x_i, Z)$ is the vector of covariances between the i-th data point and inducing points.

In the `svi` method we directly optimize this ELBO with respect to both variational parameters and kernel hyper-parameters in a stochastic way. The authors of the method suggest to use the stochastic gradient descent with natural gradients for the variational parameters and usual gradients for kernel hyper-parameters.

Natural gradients are gradients with respect to the natural parameters of an exponential family of distributions. These gradients are considered to be effective in case of optimization with respect to probability distribution parameters, because they use symmetrized KL divergence between the distributions instead of usual distance between distribution parameters as a distance metric. For more information about natural gradients see, for example [5].

The complexity of computing a stochastic update of variational and kernel parameters is independent of n and scales as $\mathcal{O}(m^3)$. The complexity of one pass over data (epoch) is thus $\mathcal{O}(nm^3)$ which is worse, than the corresponding complexity of the `vi` method. However, the stochastic optimization might give this method advantage against the `vi` method, because stochastic optimization some times leads to faster convergence in big data problems.

However, for big problems the number of required inducing points m is usually quite big. The number of parameters we have to optimize scales as $\mathcal{O}(m^2)$. Indeed, we need to optimize the bound with respect to the variational parameters μ (m parameters) and Σ ($\frac{m(m+1)}{2}$ parameters), and with respect to kernel hyper-parameters. This makes the optimization problem of the `svi` method much harder than the one we have to solve in the `vi` method (where we only have to optimize the bound with respect to kernel hyper-parameters). We will compare the two methods in the experiments section.

2.4 SVI-classification method

Finally, we can apply the bound (6) to the classification problem. In this case, we can't analytically compute the expectations $\mathbb{E}_{q(f_i)} \log p(y_i|f_i)$. However, this expectations are one-dimensional Gaussian integrals and can thus be approximated with a wide range of techniques.

The method was proposed in [4]. The authors suggest to use Gauss-Hermite quadratures in order to approximate the expectations in (6) and their derivatives.

For this method stochastic optimization is applicable, which makes it suitable for big data problems.

3 Experiments

In this section we compare the methods, described above, for both regression and classification problems. We will compare the methods, that use inducing points with the standard methods and also compare the different versions of `vi` and `svi` methods to each other.

All of the plots, apart from the plots in the next subsection, have title of the following format.

$$[\text{name of the dataset}], n = [\text{number of objects in the training set}],$$

$$d = [\text{number of features}], m = [\text{number of inducing inputs}]$$

The plots in the next subsection do not have the m in the title, because this parameter is not fixed in the corresponding experiments. If the name of the dataset is "generated", it means that the dataset was sampled from some Gaussian process.

For the regression problem we use the R^2 score, which is given by

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y}_i)^2},$$

where $\bar{y}_i = \frac{1}{n} \sum_{i=1}^n y_i$. Here y is the vector of true answers on the test set, and $\hat{y}$ is the vector of predicted answers.

For the classification problem we use accuracy score.

In all the experiments we used the squared exponential covariance function.

3.1 Inducing input methods and standard methods

We've seen above, that inducing input methods have a much smaller computational complexity than the standard methods for both GP-regression and GP-classification problems. In this section we empirically compare the prediction quality on the test data for these methods with the quality obtained by the standard methods.

We also explore the dependence between the number of inducing points used by the method and the prediction quality.

For the regression problem we use two variations of the `vi` method. `vi-means` method does not maximize the lower bound with respect to the positions of inducing inputs and just uses the K-Means cluster centers as the positions of inducing inputs. The `vi` method on the other hand does optimize for the inducing input positions. `full GP` method is the standard GP-regression method, described in section 1.2.

Fig. 5 shows the dependence between the prediction quality and the number of inducing inputs for `vi` methods for regression and classification problems.

As we can see, optimization with respect to the positions of inducing inputs does not dramatically increase the quality of predictions in the provided experiment. It does however make the optimization problem that we have to solve much harder. In general, the optimization of inducing input positions does increase the prediction quality, but makes the method much slower. We will thus abandon this method and only use `vi-means` in other experiments.

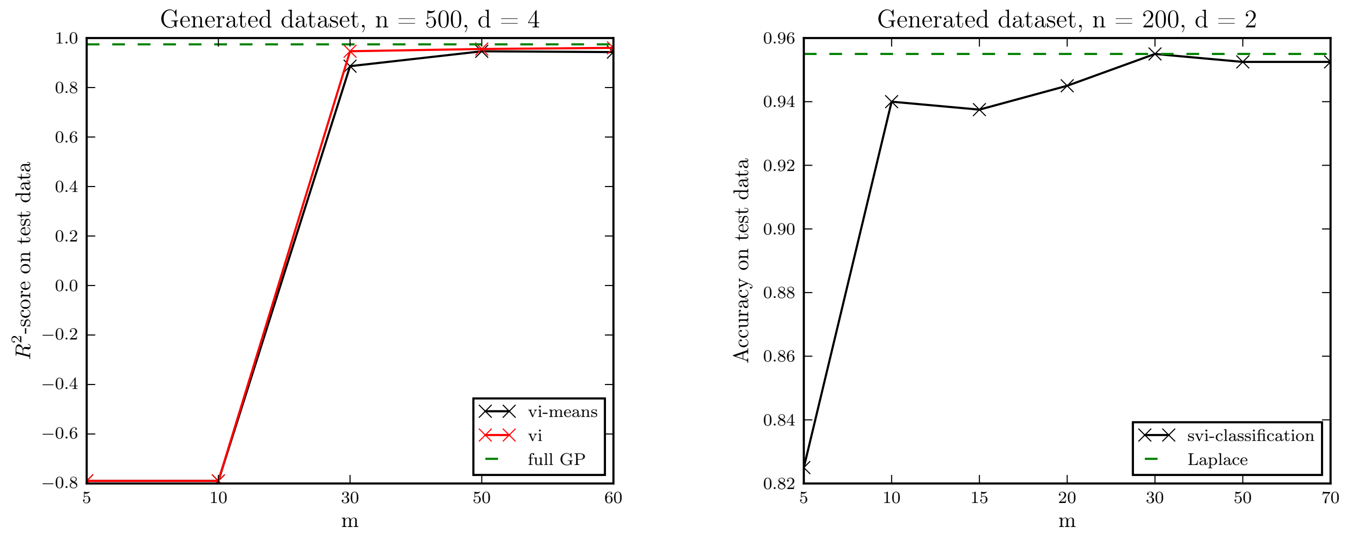


Fig. 5: The dependence between the prediction quality and the number of inducing inputs for standard and inducing point methods

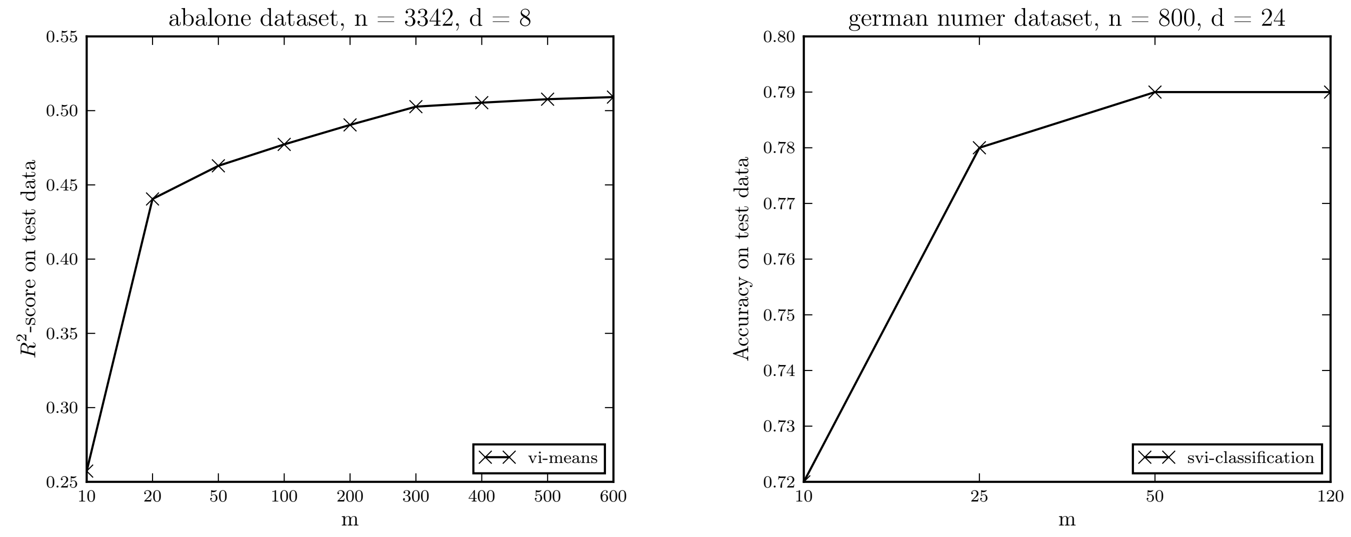


Fig. 6: The dependence between prediction quality and the number of inducing inputs for `vi-means` and `svi-classification` methods

Fig. 6 shows the dependence between the prediction quality and the number of inducing inputs for lstinlinevi-means and `svi-classification` methods. As we can see, the prediction quality gets better as the number of inducing inputs grows. However, it's hard to say, how many inducing points one should use in practice. The best answer is probably the biggest amount one can afford to train.

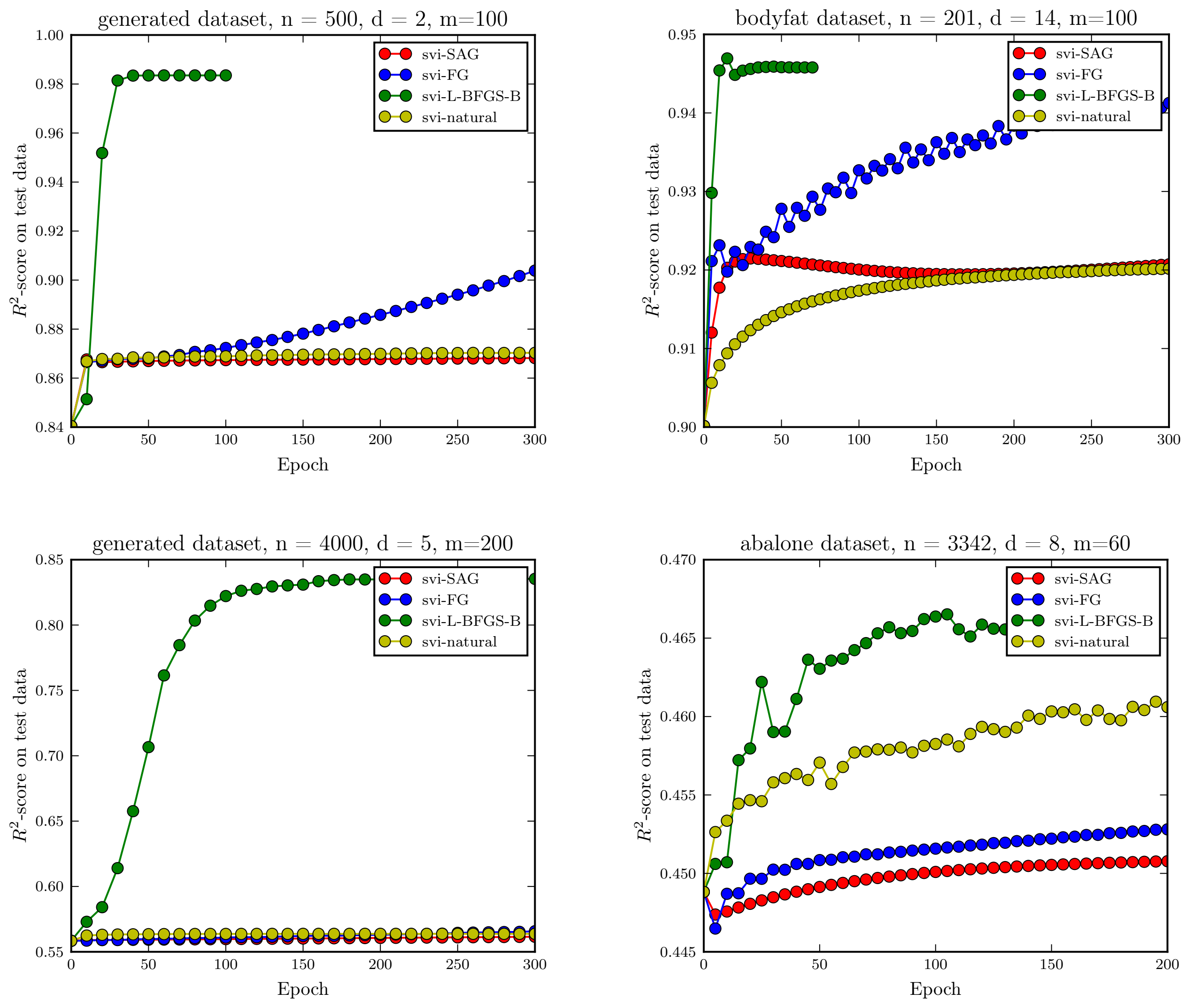


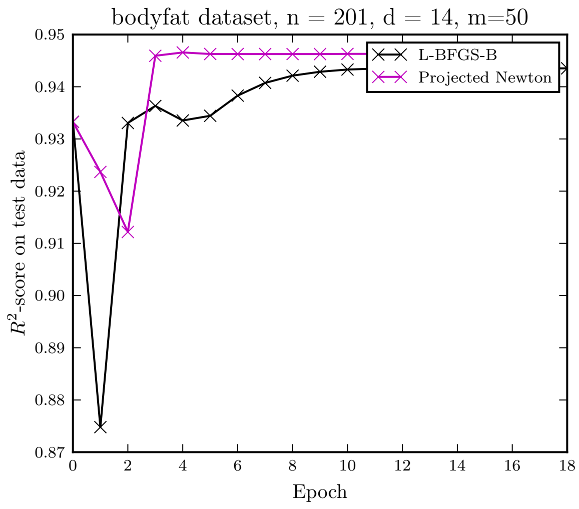
Fig. 7: `svi` methods' performance on small and medium datasets

3.2 SVI method variations

In this section we compare several variations of the `svi` method for the regression problem.

The first variation is denoted by `svi-natural`. It is the method as it was proposed in [3]. It uses stochastic gradient descent with natural gradients for minimizing the ELBO with respect to the variational parameters, and usual gradients with respect to kernel hyper-parameters.

The methods `svi-L-BFGS-B` and `svi-FG` use the same lower bound (8) and optimize it with deterministic optimization methods L-BFGS-B and projected gradient respectively. We use the bound-constrained optimization methods, because the hyper-parameters of the squared exponential kernel must be positive.



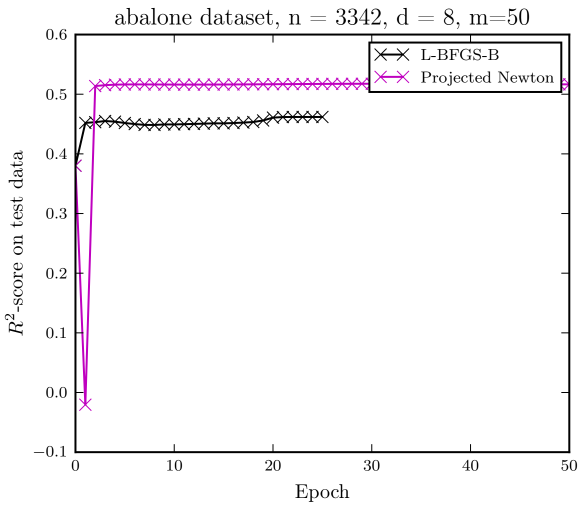


Fig. 8: `vi` method variations on different datasets

We can not use the natural gradients in this setting, because they are not necessarily a descent direction and can't be used by L-BFGS-B or gradient descent. Thus, we use usual gradients with respect to variational parameters μ and Σ for these methods. However, the matrix Σ has to be symmetric and positive definite and we have to ensure that our optimization updates maintain these properties. In order to avoid complex constrained optimization problems, we use Cholesky decomposition of Σ and optimize the bound with respect to the Cholesky factor L_Σ of Σ. This allows us to solve a simpler bound-constrained problem instead of a general constrained optimization problem.

Finally, the `svi-SAG` uses stochastic average gradient method to minimize the ELBO. This method also uses Cholesky factorization and usual gradients instead of natural for the same reasons. For more information about SAG method see [7].

The results on small and medium datasets are shown in fig. 7.

As we can see, on these moderate problems using stochastic optimization does not give any advantages against the L-BFGS-B method. However, using the natural gradients allows the stochastic gradient descent method to beat SAG. For these reasons we will only use the `svi-natural` and `svi-L-BFGS-B` methods in the comparison with the `vi-means` method.

3.3 VI method variations

In this section we compare two optimization methods for the `vi-means` method.

The first variation is denoted by `Projected Newton`. It uses projected Newton method for minimizing the ELBO (7). The second variation is denoted by `means-L-BFGS-B` and uses L-BFGS-B optimization method.

`Projected Newton` method uses finite-difference approximation of the hessian. It also makes hessian-correction in order to make it symmetric and positive-definite. The optimization method itself makes a Newton step and then projects the result to the feasible set in the metric, determined by the hessian. For more information about the method see for example [6].

The time complexity of one iteration for both projected Newton method and L-BFGS-B is $\mathcal{O}(nm^2)$. In the projected Newton method we have to compute the hessian matrix of the ELBO with respect to covariance hyper-parameters. In case of squared exponential covariance function the time, needed to compute the hessian, is twice the time, needed to compute the gradient.

The results are provided in fig. 8. In the provided experiments projected Newton method beats L-BFGS-B. However, the results are close and on different datasets L-BFGS-B beats projected Newton. We need to perform further experiments in order to find out whether using second order optimization provides benefits in the `vi` method.

In further experiments we use the L-BFGS-B method because it was more stable in general in our experiments.

3.4 Comparison of VI and SVI methods

In this section we compare the `vi-means` method (with L-BFGS-B optimization method) with `svi-L-BFGS-B` and `svi-natural`. We've described these methods above.

We've seen, that the computational complexity of one epoch of the `vi-means` method is better method than it is for the `svi-natural` and `svi-L-BFGS-B`. However, the `svi-natural` method uses stochastic optimization, which might lead to faster convergence in big data problems.

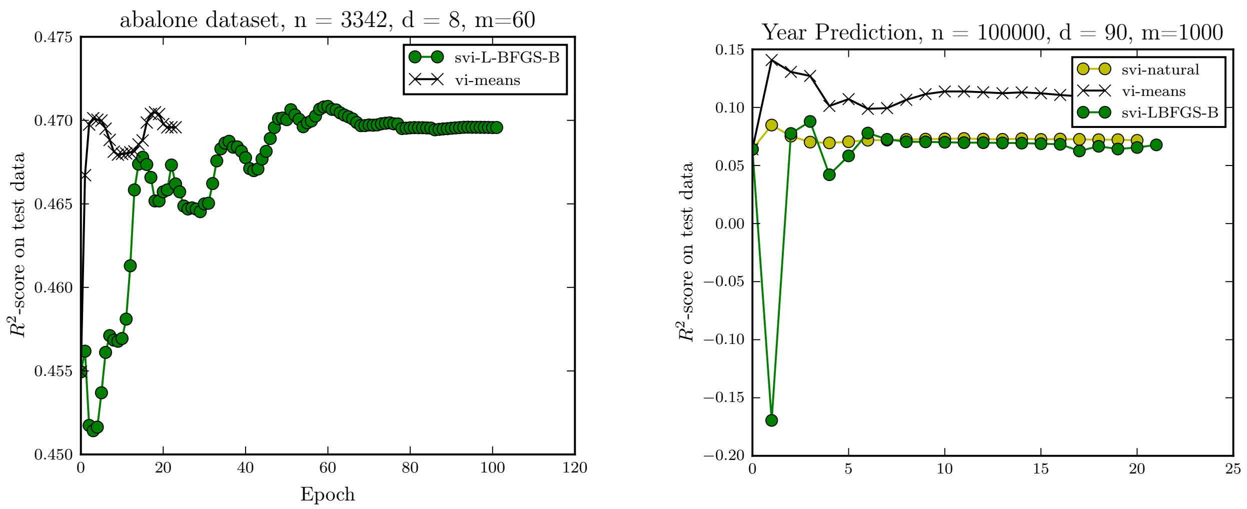


Fig. 9: `vi` and `svi` methods comparison

Fig. 9 shows the results of the experimental comparison of `vi` and `svi` methods. In the first experiment of these two we didn't run the `svi-natural`, because we've already compared it to the `svi-L-BFGS-B` method on this exact dataset and it proved to be worse (see fig. 7).

We can see, that `svi-natural` performs slightly better, then the deterministic `svi-L-BFGS-B`, but can't beat `vi-means`. The reason for that is that the optimization problem for the `svi` method is much harder then the one, solved by the `vi` method. Indeed, for $m = 1000$ and squared exponential kernel we have about $5 \cdot 10^5$ optimization parameters for `svi` methods and just 3 parameters for the `vi` method.

4 Conclusions

As a result of the experiments, we can make several conclusions.

First of all, the inducing input methods proved to be effective. They allow applying the gaussian process framework to the big data problems, which can not be done with the standard methods.

The `vi` method seems to work better than the `svi` method even for big data problems, despite the fact, that one can not use stochastic optimization with it. One of the reason for that is that the optimization problem itself in the `vi` method is much easier than the one, that has to be solved in the `svi` method, and using stochastic optimization does not really help.

The stochastic average gradient optimization method for the `svi` method does not beat the stochastic gradient descent with natural gradietns. This implies, that using the natural gradients instead of the usual gradients in the `svi` method provides benefits.

5 Future work

As we have seen, for the regression problem `vi` method beats `svi` in all of the provided experiments. We think that the main reason for that is the complexity of optimization with respect to variational distribution parameters μ and Σ. In the classification problem, there is no known analogy for the `vi` method. Moreover, the `svi-classification` method does not use natural gradients with respect to variational parameters. This might lead to slow covergence, as natural gradients proved to be useful in our experiments.

This leads us to the idea, that if we could avoid the optimization with respect to variational parameters μ and Σ, we might be able to obtain a faster converging method. If we use the logistic likelihood function $\log p(y_i|f_i) = -(1+\exp(-y_i f_i))$ for the classification, the problem of maximizing the lower bound is very similar to the Bayesian logistic regression problem. Paper [8] provides a quadratic lower bound for the Bayesian logistic regression setting. Using this bound in the GP-classification problem would lead to analytical formulas for optimal μ and Σ. We will compare this approach to the `svi-classification` approach.

We will also perform more experiments to determine whether using second-order optimization for the `vi` method is beneficial over L-BFGS-B.

Finally, in the `svi-classification` method we can use the gradient of one sample $\log p(y_i|\tilde{f})$ from the distribution $\tilde{f} \sim q(f_i)$ to onbtain an unbiased estimate of the gradient of the expectation $\mathbb{E}_{q(f_i)} \log p(y_i|f_i)$. Now we use Gauss-Hermite quadratures to approximate the expectation. This might lead to a faster version of the `svi-classification` method.

Literature

[1] Rasmussen, C. E. and Williams, C. K. I. (2006). Gaussian Processes for Machine Learning. *MIT Press.*

[2] Titsias M. K. (2009). Variational Learning of Inducing Variables in Sparse Gaussian Processes. In: *International Conference on Artificial Intelligence and Statistics*, pp. 567–574.

[3] Hensman J., Fusi N., Lawrence D. (2013). Gaussian Processes for Big Data. In: *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence.*

[4] Hensman J., Matthews G., Ghahramani Z. (2015). Scalable Variational Gaussian Process Classification. In: *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics.*

[5] Efron B. (1978). The Geometry of Exponential Families. In: *The Annals of Statistics.*

[6] Schmidt M., Kim D., Sra S. (2011). Projected Newton-Type Methods in Machine Learning. In: *Optimization for Machine Learning.*

[7] Schmidt M., Le Roux N., Bach F. (2013). Minimizing Finite Sums with the Stochastic Average Gradient. *Manuscript.*

[8] Jaakkola T., Jordan M. (1996). A Variational Approach to Bayesian Logistic Regression Models and Their Extensions. In: *Artificial Intelligence and Statistics.*

[9] Quinonero-Candela, J. and Rasmussen, C. (2005). A Unifying View of Sparse Approximate Gaussian Process Regression. In: *Journal of Machine Learning Research*

[10] Chalupka K., Williams C., Murray I. (2013). A Framework for Evaluating Approximation Methods for Gaussian Process Regression. In: *Journal of Machine Learning Research*

[11] Jaakkola, T. (2001). Tutorial on Variational Approximation Methods. In: *M. Opper and D. Saad (Eds.), Advances in Mean Field Methods*