LOMONOSOV MOSCOW STATE UNIVERSITY
FACULTY OF COMPUTATIONAL MATHEMATICS AND CYBERNATICS
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

1 Introduction

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.

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 it's 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.

In the fig. 1 an example of a one-dimensional Gaussian process is provided. The darker blue line is the mean of the process, and the lighter blue region is the 3σ-region, drawn at each point.

1.2 Gaussian process regression

Consider the regression problem. We have a dataset $\{(x_i, f_i) | i = 1, \ldots, n\}$, 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 $x_1, \ldots, x_n$ 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 $X_* \in \mathbb{R}^{l \times d}$.

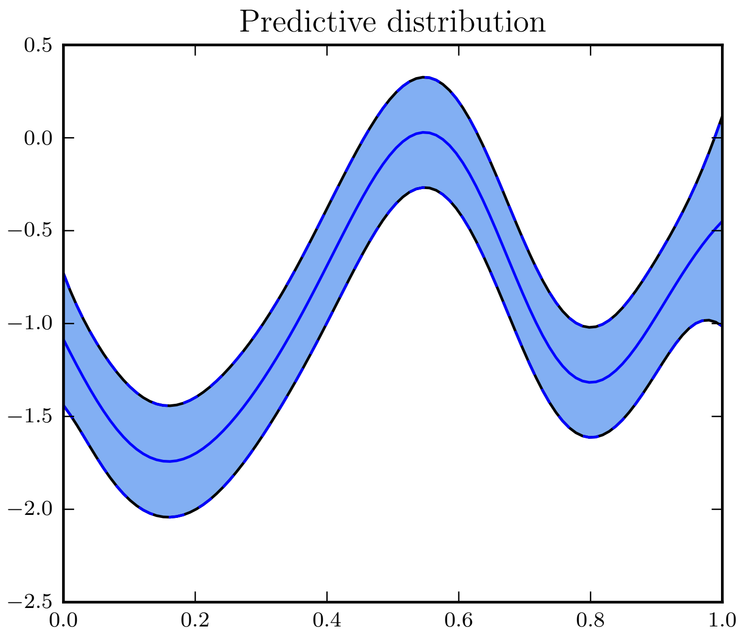


Fig. 1: One-dimensional Gaussian processes

We will also 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

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

for some noise variance $\sigma_n \in \mathbb{R}$. 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

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

Then, the joint distribution of f and f_* is given by

$$\left[\begin{array}{c} f \\ f_* \end{array} \right] \sim \mathcal{N}\left(0, \left[\begin{array}{cc} K(X, X) & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{array} \right]\right).$$

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

$$\left[\begin{array}{c} y \\ f_* \end{array} \right] \sim \mathcal{N}\left(0, \left[\begin{array}{cc} K(X, X) + \sigma_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{array} \right]\right).$$

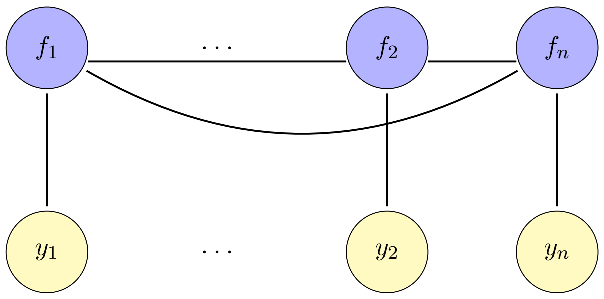


Fig. 2: The graphical model for Gaussian process regression and classification

Conditioning this distribution, we obtain the predictive

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

where

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

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

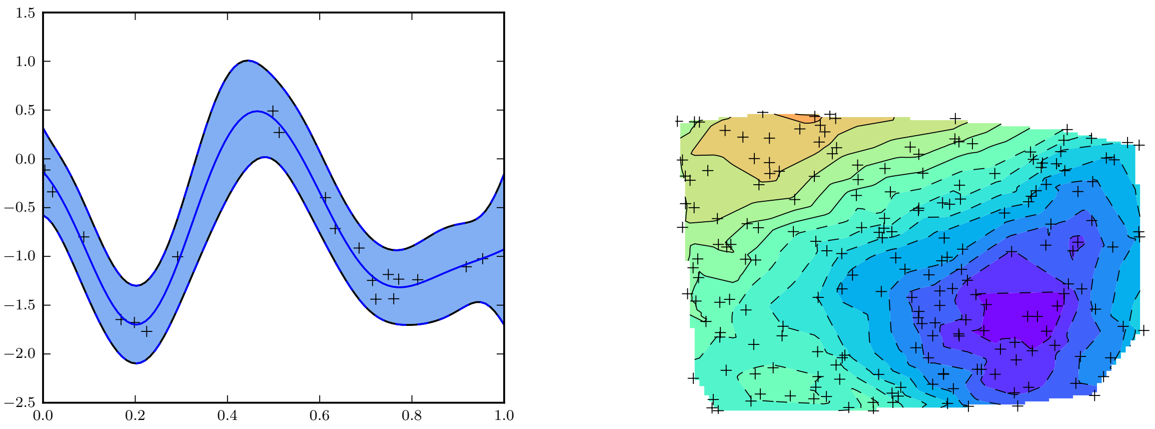


Fig. 3: One and two-dimensional Gaussian processes, reconstructed from data

In fig. 3 you can see the examples of one and two-dimensional Gaussian processes, reconstructed from the 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 $\{(x_i, y_i)|i = 1, \ldots, n\}$, where $x_i \in \mathbb{R}^d$, $y_i \in \{-1, 1\}$.

We want to predict the probabilities of new datapoints x_* belonging to positive class.

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

$$p(f_*|X, y, x_*) = \int p(f_*|X, x_*, f)p(f|X, y)df. \tag{1}$$

Now, the probability of the positive class is given by marginalizing over the latent variable f_*.

$$p(y_* = +1|X, y, x_*) = \int \sigma(f_*)p(f_*|X, y, x_*)df_*. \tag{2}$$

Unfortunantely, both the integrals in (1) and (2) 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 it's maximum $\hat{f}$.

Substituting this Gaussian approximation back into (1) and (2), 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 the previous two sections, we described, how to fit a Gaussian process to the data in the regression and in the 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

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

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.

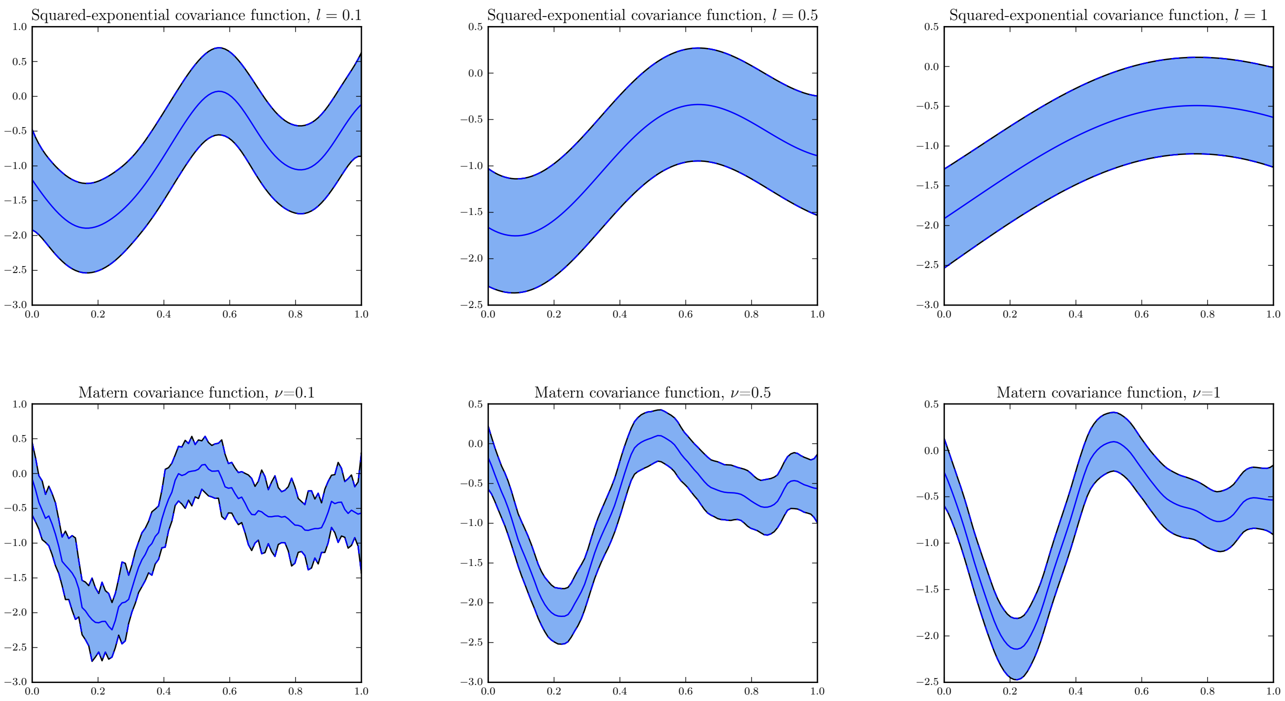


Fig. 4: Gaussian processes with squared-exponential and Matern covariance functions, reconstructed from the same data for different values of hyper-parameters

In fig. 4 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 maximizization of the evidence, or marginal likelihood of the model. Marginal

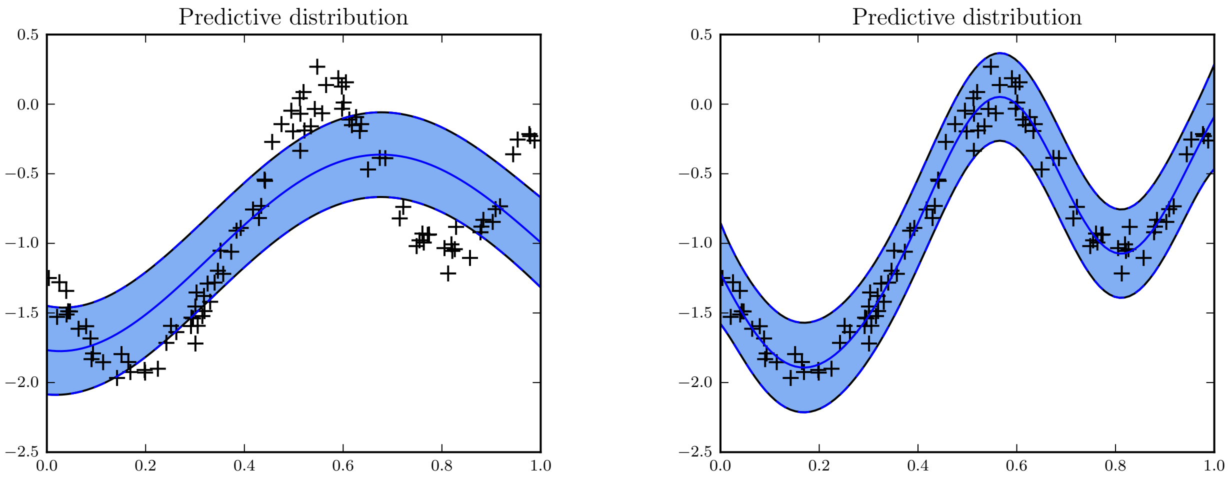


Fig. 5: Predictive distribution before and after hyper-parameter adaptation

likelihood is given by

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

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

$$\log p(y|X) = -\frac{1}{2}y^T(K + \sigma_n^2 I)^{-1}y - \frac{1}{2}\log|K + \sigma_n^2 I| - \frac{n}{2}\log 2\pi. \tag{3}$$

For the laplace approximation method, the marginal likelihhod is also available in the closed form.

Thus, for the regression problem, we should first maximize the evidence, given by (3) 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].

Fig. 5 provides an example of the GP-regression predictive distribution for the same dataset before and after tuning the kernel hyper-parameters. It can be seen from the plots, that the model with tuned hyper-parameters, describes the data much better.

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.

We will consider the approach, based on the concept of inducing inputs. 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

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

The corresponding graphical model is shown in fig. 6. 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

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

The points Z are called the 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.

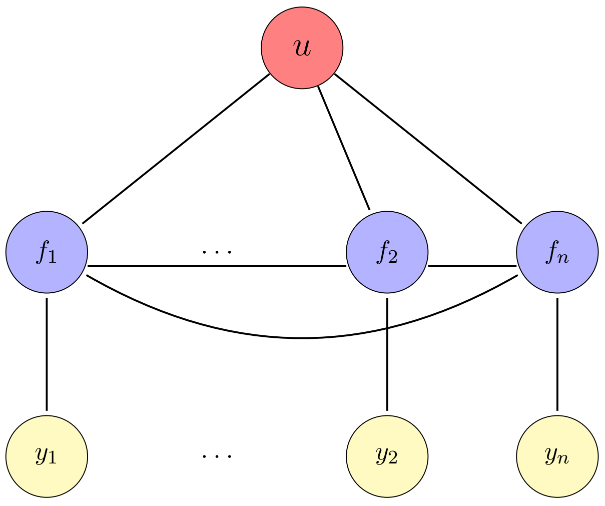


Fig. 6: The augmented model

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

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

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

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

Applying the standard variational lower bound to the augmented model, we obtain the following inequality.

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

for any distribution $q(u, f)$. 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

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

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,

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

Combining everything back together, we obtain the evidence lower bound

$$\log p(y) \geq \sum_{i=1}^{n} \mathbb{E}_{q(f_i)} \log p(y_i|f_i) - \mathrm{KL}\left(q(u) \mid\mid p(u)\right). \tag{4}$$

This lower bound 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 (4).

2.2 VI method

The first method we will describe was introduced in [2]. In the case of the regression problem we can analytically optimize the bound (4) with respect to variational parameters. The optimal values are

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

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

Now, substituting the optimal values of variational parmeters back into 4, we obtain a new lower bound

$$\log p(y) \geq \log \mathcal{N}(y | 0, \sigma_n^2 I + K_{nm} K_{mm}^{-1} K_{mn}) - \frac{1}{2\sigma_n^2} \mathrm{tr}(\tilde{K}), \tag{5}$$

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

This lower bound and it's derivatives with respect to covariance hyper-parameters can be computed in $\mathcal{O}(nm^2)$ operations. 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 the 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 (4) 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 (5) 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 (4) does have a form of sum over objects, and we can thus apply stochastic methods to it. In the regression case the bound looks like

$$\log p(y) \geq \sum_{i=1}^{n} \left(\log \mathcal{N}(y_i | k_i^T K_{mm}^{-1} \mu, \sigma_n^2) - \frac{1}{2\sigma_n^2} \tilde{K}_{ii} - \frac{1}{2} \mathrm{tr}(\Sigma \Lambda_i) \right) -$$
$$- \frac{1}{2} \left(\log \frac{|K_{mm}|}{|\Sigma|} - m + \mathrm{tr}(K_{mm}^{-1} \Sigma) + \mu^T K_{mm}^{-1} \mu \right). \tag{6}$$

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.

The natural gradients are the gradients with respect to the natural parameters of an exponential family of distributions. These gradients are considered to be effective in the 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 the variational and kernel parameters is independent of n and scales as $\mathcal{O}(m^3)$. Thus, the stochastic optimization might give this method advantage against the `vi` method. However, for big data problems the number of required inducing points m is usually quite big. The number of parameters we have to optimize scales as $\mathcal{O}(m^2)$, which makes the optimization problem of the `svi` method much harder then the one we have to solve in the `vi` method. We will compare the two methods in the experiments section.

2.4 SVI-classification method

Finally, we can apply the bound (4) 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 (4) 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

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

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 then 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. The `vi-means` method does not maximize the lower bound with respect to the positions of the inducing inputs and just uses the K-Means cluster centers as the positions of the 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.

In the two top plots of fig. 7 the dependence between the number of inducing points and the prediction quality is shown on two small datasets. As we can see, the optimization with respect to the positions of the inducing inputs does not dramatically increase the quality of the predictions. It does however make the optimization problem that we have to solve much harder and makes the method much slower. We will thus abandon this method and only use `vi-means` method in other experiments.

We can also see from these plots that for sufficient number of inducing inputs the `vi` methods reach the predictive quality of the standard methods.

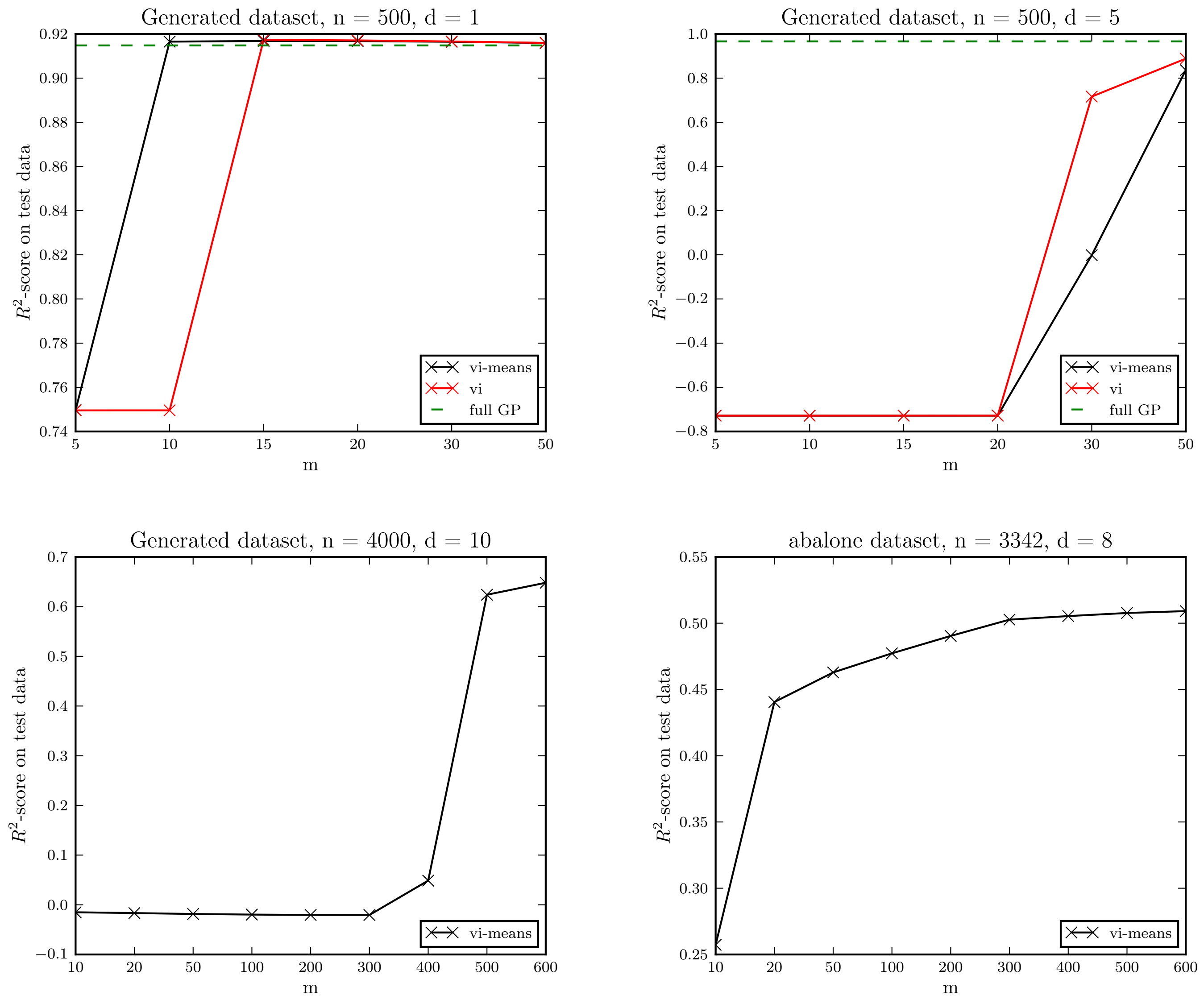
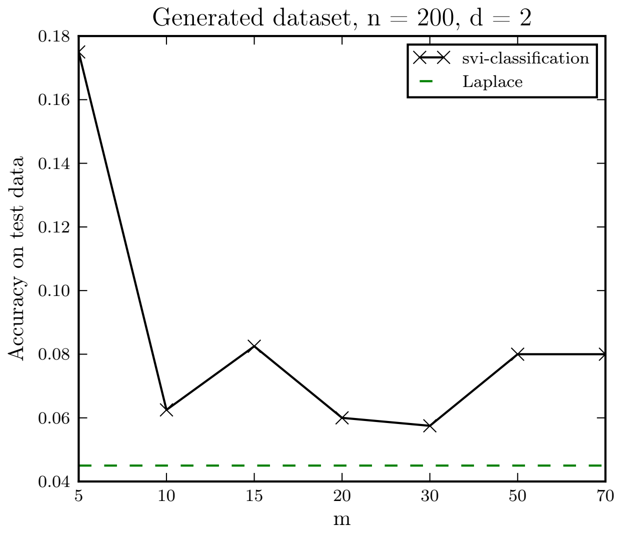


Fig. 7: The dependence between prediction quality and the number of inducinge inputs for the regression problem

In the bottom plots of fig. 7, the dependence between the number of used inducing inputs and the quality of the `vi-means` method predictions is shown for two bigger datasets.

In fig. 8 the results of similar experiments for the classification problem are provided. Here we compare the `Laplace` method, which was described in section 1.3, and the `svi-classification` method, which was described in section 2.4.

As we can see, on the chosen dataset the `svi-classification` method can not reach the predictive quality of the `Laplace` method even for reasonably big values of m. However, the time consumption of the `Laplace` method does not allow to use it even for moderate problems.



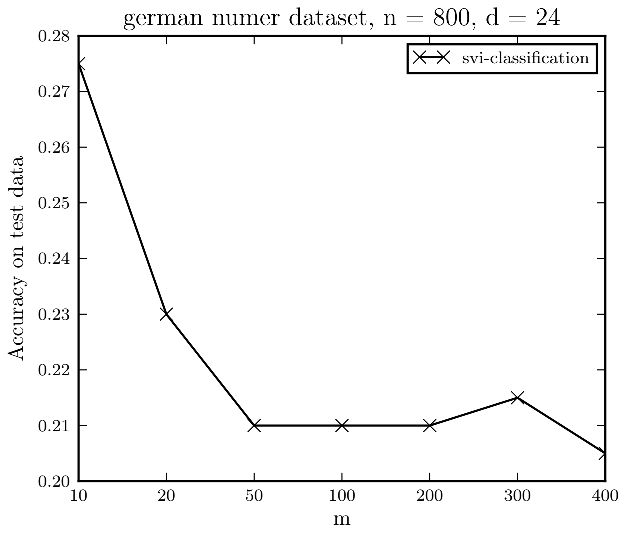


Fig. 8: The dependence between prediction quality and the number of inducinge inputs for the classification problem

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 (6) and optimize it with deterministic optimization methods L-BFGS-B and gradient descent respectively.

We can not use the natural gradients in this setting, because they are not necesserily a good direction for optimization 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, which leads to a bound-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.

The results on small and medium datasets are shown in fig. 9.

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

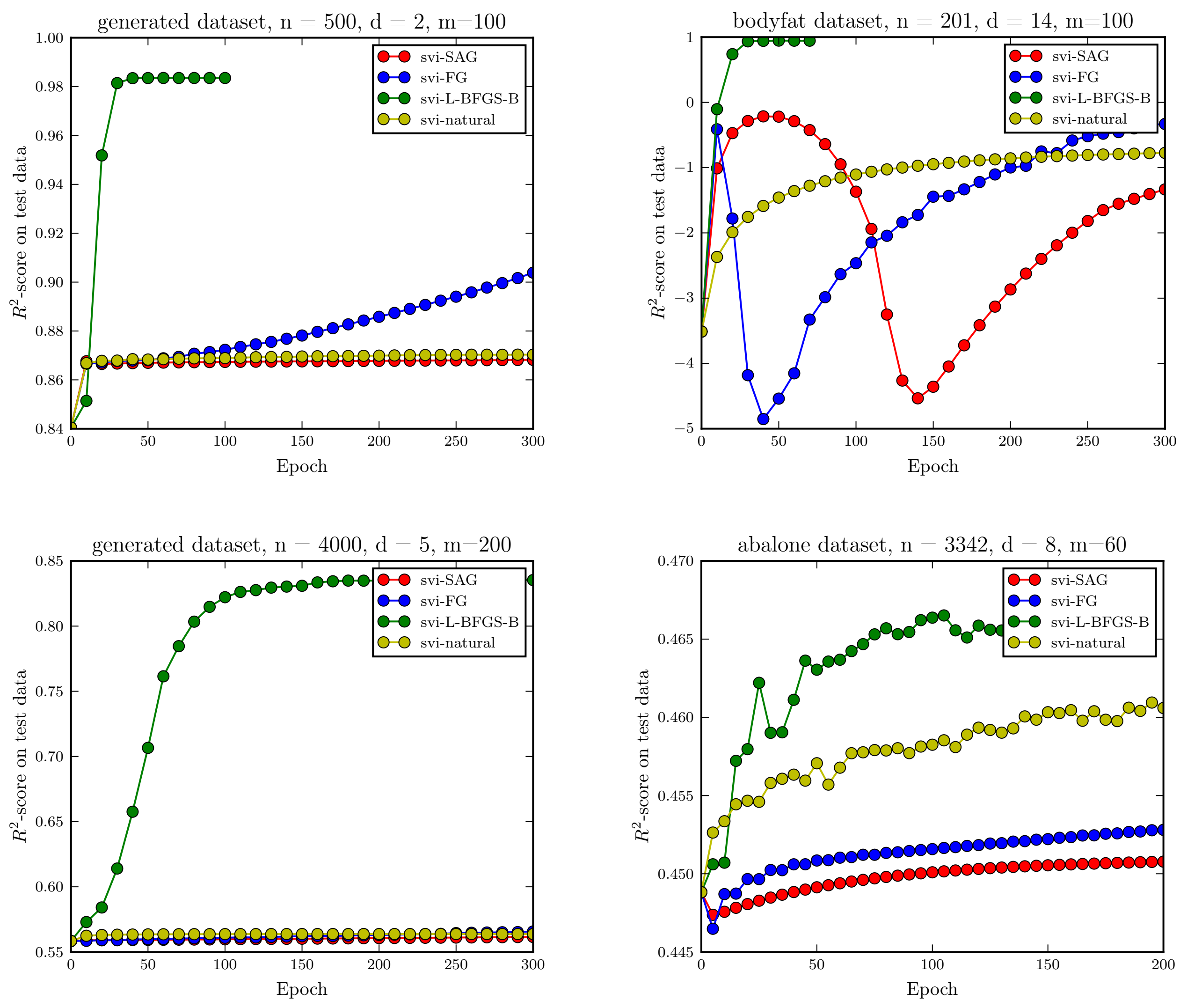


Fig. 9: Svi methods' performance on small and medium datasets

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 the 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 (5). The second variation is denoted by `means-L-BFGS-B` and uses L-BFGS-B optimization method.

The `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

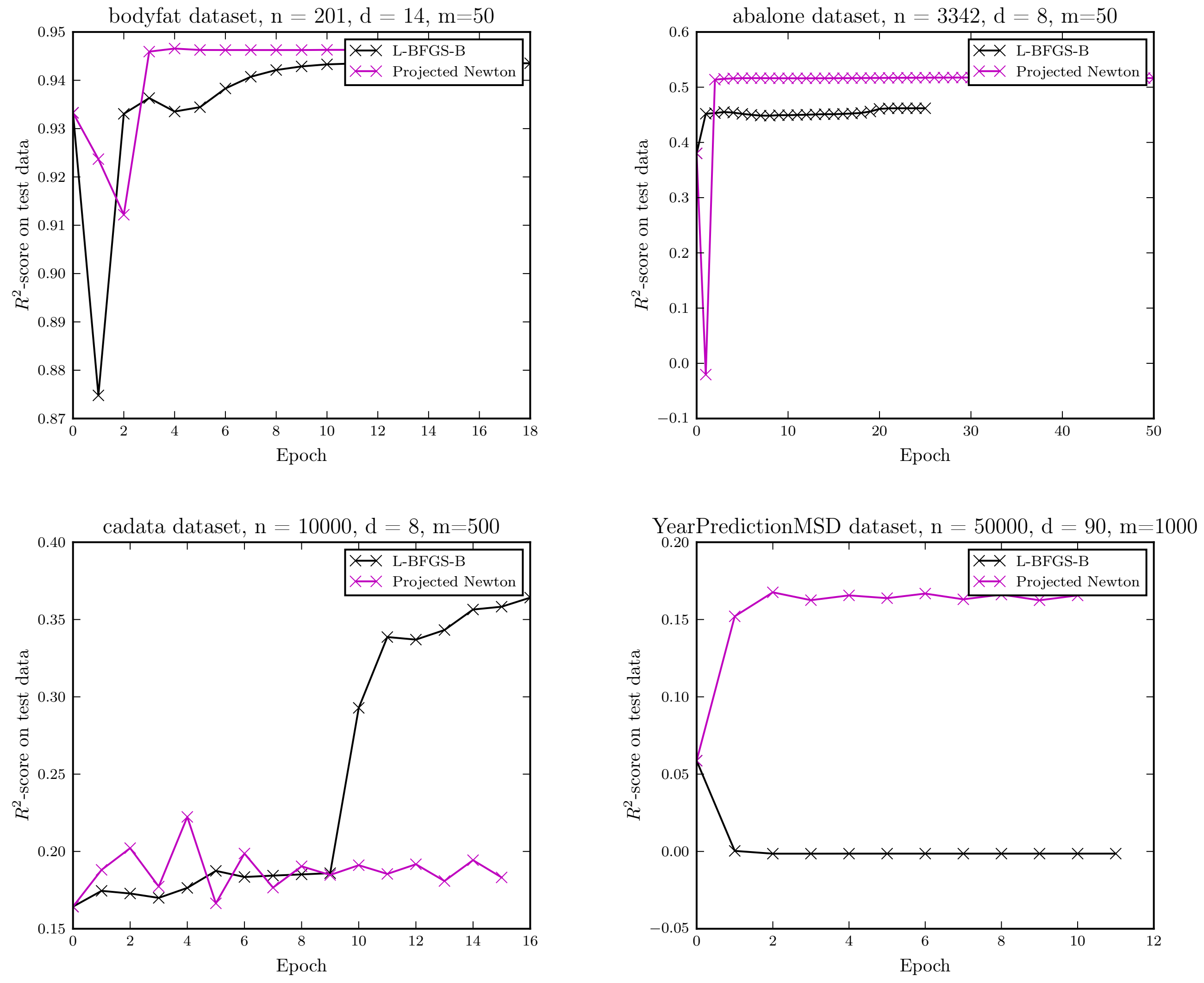


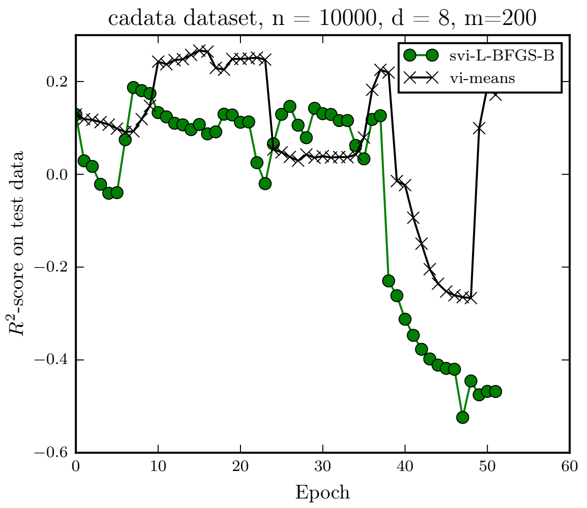
Fig. 10: Method's performance on a bigger dataset

set in the metric, determined by the hessian. For more information about the method see for example [6].

The results are provided in fig. 10. As we can see, on different datasets the results are different and it's hard to say, which of the methods is better. In the big experiment the methods seem to have converged to different optimums. In the further experiments we use the L-BFGS-B method.

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 this methods above.



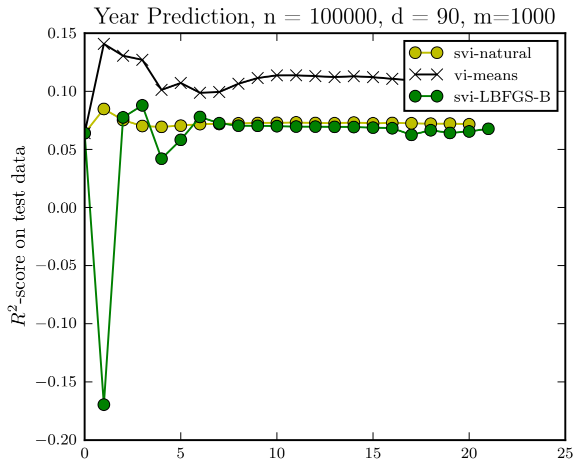


Fig. 11: VI and SVI methods comparison

The results are provided in fig. 11. 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. 9).

We can see, that `vi-means` beats `isvi` in all the experiments. One could expect these results, because `vi-means` optimizes the exact same functional as it's oponent, but it uses exact optimal values for some of the parameters. However, the `svi-natural` method uses stochastic optimization which should help it in big data problems. We can see, that it performs slightly better, then the deterministic `svi-L-BFGS-B`, but can't beat the `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 gradietns in the `svi` method provides benifits.

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.*