Optimization of Gaussian Process Hyperparameters using Rprop

Manuel Blum and Martin Riedmiller

University of Freiburg - Department of Computer Science
Freiburg, Germany

Abstract. Gaussian processes are a powerful tool for non-parametric regression. Training can be realized by maximizing the likelihood of the data given the model. We show that Rprop, a fast and accurate gradient-based optimization technique originally designed for neural network learning, can outperform more elaborate unconstrained optimization methods on real world data sets, where it is able to converge more quickly and reliably to the optimal solution.

1 Gaussian Process Regression

Gaussian processes (GP) are defined as a finite collection of jointly Gaussian distributed random variables. For regression problems these random variables represent the values of a function $f(\boldsymbol{x})$ at input points $\boldsymbol{x}$. Prior beliefs about the properties of the latent function are encoded by the *mean function* $m(\boldsymbol{x})$ and *covariance function* $k(\boldsymbol{x}, \boldsymbol{x}')$. Thereby, all function classes that share the same prior assumptions are covered and inferences can be made directly in function space.

In order to make predictions based on data, we consider the joint Gaussian prior of the noisy training observations $\boldsymbol{y}$ and the test outputs $\boldsymbol{f}_*$. We derive the posterior distribution by conditioning the prior on the training observations, such that the conditional distribution of $\boldsymbol{f}_*$ only contains those functions from the prior that are consistent with the training data. Assuming the prior mean to be zero, the following predictive equations for GP regression are obtained:

$$\bar{\boldsymbol{f}}_* = \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}) \left[\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \boldsymbol{I}\right]^{-1} \boldsymbol{y} \tag{1}$$

$$\operatorname{cov}(\boldsymbol{f}_*) = \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}_*) - \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}) \left[\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \boldsymbol{I}\right]^{-1} \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*) \tag{2}$$

$\boldsymbol{K}$ is the kernel matrix, $\boldsymbol{X}$ and $\boldsymbol{X}_*$ are the training inputs and the test inputs respectively, $\boldsymbol{y} = f(\boldsymbol{x}) + \epsilon$ is the vector of training observations, where ϵ is additive noise which is assumed to be Gaussian distributed with zero-mean and variance σ_n^2. The kernel matrix $\boldsymbol{K}$ is constructed by evaluating the covariance function between all pairs of inputs points. See [1] for further details on Gaussian processes.

1.1 Model selection and hyperparameters

A GP is fully specified by its mean function $m(\boldsymbol{x})$ and covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$. Usually, the mean function is fixed to zero, which is not a strong limitation if the data is centered in preprocessing. The covariance function defines similarity between data points and is chosen such that it reflects the prior beliefs about the function to be learned. Every kernel function that gives rise to a positive semidefinite kernel matrix can be used.

One of the most commonly employed kernels for GPs is the squared exponential covariance function $k_{SE}(x, x') = \sigma_f^2 \cdot \exp\left(-\frac{\|x-x'\|^2}{2l^2}\right)$, which reflects the prior assumption that the function to be learned is smooth. The parameter l is called the characteristic length-scale and specifies, roughly speaking, the distance from which on two points will be uncorrelated. Parameter σ_f^2 controls the overall variance of the process.

The free parameters, called *hyperparameters*, allow for flexible customization of the GP to the problem at hand. The choice of the covariance function and its hyperparameters is called *model selection.* In the following, we will use the Bayesian view on model selection, where the optimal hyperparameters are determined by maximizing the probability of the model given the data.

1.2 Marginal likelihood

For Gaussian process regression with Gaussian noise it is possible to obtain the probability of the data given the hyperparameters $p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})$ by marginalization over the function values $\boldsymbol{f}$. The log marginal likelihood is given in Eq. 3,

$$\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = -\frac{1}{2}\boldsymbol{y}^T \boldsymbol{K}_y^{-1} \boldsymbol{y} - \frac{1}{2}\log|\boldsymbol{K}_y| - \frac{n}{2}\log 2\pi \tag{3}$$

where $\boldsymbol{K}_y = \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \boldsymbol{I}$ is the covariance function for the noisy targets $\boldsymbol{y}$. The first term in Eq. 3 can be interpreted as a data-fit term, the second term is a complexity penalty and the last term is a normalizing constant. The derivatives of the log marginal likelihood with respect to the hyperparameters are given by:

$$\frac{\partial}{\partial \theta_j} \log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = \frac{1}{2}\mathrm{tr}\left(\left(\boldsymbol{\alpha}\boldsymbol{\alpha}^T - \boldsymbol{K}_y^{-1}\right)\frac{\partial \boldsymbol{K}_y}{\partial \theta_j}\right) \tag{4}$$

Using Eq. 4 any gradient based optimization algorithm can be used to obtain the hyperparameters that maximize the marginal likelihood of a GP. We will call this optimization procedure *training* the GP.

2 The Rprop algorithm

Rprop is a fast and accurate gradient-based optimization technique originally designed for neural network learning [2], but it has been successfully applied to a variety of problems, including robot localization [3], motion planning [4] and traffic control [5]. In contrast to *gradient descent*, which finds a local minimum of a function $J(\boldsymbol{\theta})$ by taking iterative steps proportional to the negative local gradient, Rprop uses adaptive update steps, which only depend on the sign of the gradient:

$$\theta_i^{(t+1)} = \theta_i^{(t)} - \operatorname{sign}\left(\frac{\partial J}{\partial \theta_i}^{(t)}\right)\Delta_i^{(t)} \tag{5}$$

If the sign of partial derivative of J with respect to parameter θ_i stays the same, the update-value $\Delta\theta_i$ is increased by a factor $\eta^+ > 1$ in order to accelerate convergence. If the sign of the derivative changes, the update-value Δ_i is decreased by the factor $0 < \eta^- < 1$ and adaption in the succeeding learning step is inhibited by setting $\frac{\partial J}{\partial \theta_i}^{(t-1)}$ to zero.

$$\Delta_i^{(t)} = \begin{cases} \eta^+ \cdot \Delta_i^{(t-1)}, & \text{if } \frac{\partial J}{\partial \theta_i}^{(t-1)} \cdot \frac{\partial J}{\partial \theta_i}^{(t)} > 0 \\ \eta^- \cdot \Delta_i^{(t-1)}, & \text{if } \frac{\partial J}{\partial \theta_i}^{(t-1)} \cdot \frac{\partial J}{\partial \theta_i}^{(t)} < 0 \\ \Delta_i^{(t-1)}, & \text{else} \end{cases} \tag{6}$$

The update-values are initially set to Δ_0 and are bounded by Δ_{min} and Δ_{max} during the learning process. Combined with η^+ and η^- there are five parameters to be specified, but there are parameter settings that turn out to work reliably for a wide variety of problems.

Rprop is a general method for unconstrained optimization and can be used for any once-differentiable function. The application to hyperparameter optimization of Gaussian processes is straightforward: The likelihood of the data given the model is maximized by minimizing the negative log-likelihood using the gradient in Eq. 4. In the following section we will show the effectiveness of this approach and compare the results to conjugate gradients and quasi-Newton methods, the most commonly used techniques for Gaussian process hyperparameter optimization.

3 Experiments

Experiments were performed using the *Gaussian Processes for Machine Learning* (GPML) toolbox [6], which includes a Polack-Ribiere conjugate gradients implementation. The toolbox also provides a Matlab interface to the L-BFGS-B algorithm [7]. We compare the performance of Rprop to these algorithms on three test cases: Randomly generated data, the Boston housing data set and the Mauna Loa CO_2 data set. All experiments were repeated 100 times using random hyperparameter initializations and we report average results.

3.1 Synthetic data

The synthetic data is sampled from a GP according to Eq. 7, where $\boldsymbol{L}$ is the Cholesky decomposition of the covariance matrix $\boldsymbol{K} = \boldsymbol{L}\boldsymbol{L}^T$ and $\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ is a vector of standard normally distributed values. It is easy to show that $\boldsymbol{y}$ has

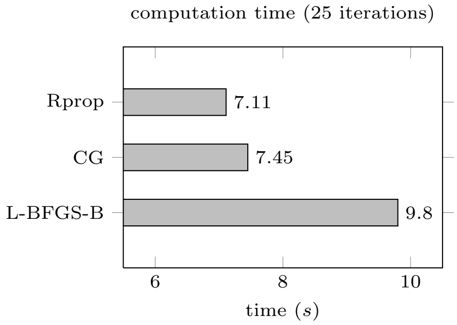

Fig. 1: Real computation time needed by the algorithms for 25 iterations.

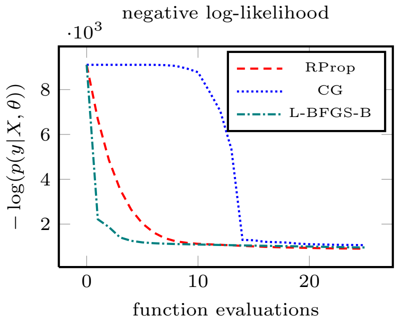

Fig. 2: Negative log-likelihood and for randomly generated 5-dimensional input.

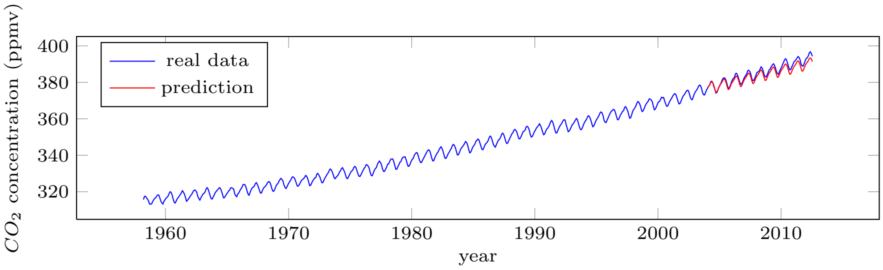

Fig. 3: Monthly CO_2 concentrations at Mauna Loa, Hawaii. The prediction (red) is based on values up to Dec 2003 using a GP learned with Rprop.

mean $m(\boldsymbol{x})$ (8) and covariance $k(\boldsymbol{x}, \boldsymbol{x}')$ (9).

$$\boldsymbol{y} = m(\boldsymbol{x}) + \boldsymbol{L}\boldsymbol{u} \tag{7}$$

$$\mathbb{E}\left[\boldsymbol{y}\right] = \mathbb{E}\left[m(\boldsymbol{x}) + \boldsymbol{L}\boldsymbol{u}\right] = m(\boldsymbol{x}) \tag{8}$$

$$\operatorname{cov}(\boldsymbol{y}) = \mathbb{E}\left[(\boldsymbol{y} - m(\boldsymbol{x}))(\boldsymbol{y} - m(\boldsymbol{x}))^T\right] = \mathbb{E}\left[(\boldsymbol{L}\boldsymbol{u})(\boldsymbol{L}\boldsymbol{u})^T\right] = \boldsymbol{K} \tag{9}$$

We randomly pertubate the hyperparameters of the Gaussian process and train it using marginal likelihood maximization. This way, mean and covariance functions are perfectly suited for the data and the optimal hyperparameters should be close to those that were used to generate the training set.

3.2 Housing data set

The housing data set contains housing values in the suburbs of Boston, as well as attributes like crime rate or nitric oxides concentration. In total there are 13 real valued attributes, which are used to predict the values of the homes. The data set is split into 455 training instances and 51 test cases. We model the

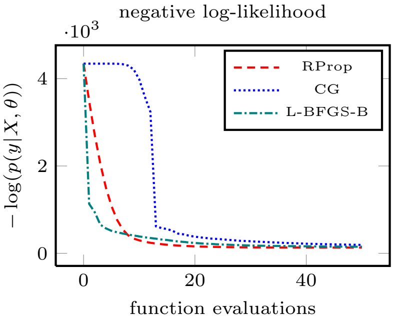

Fig. 4: Log-likelihood training curve using the Boston housing data set.

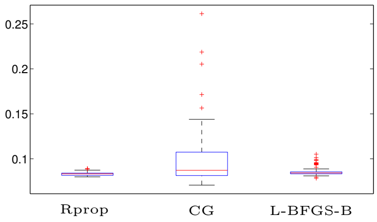

Fig. 5: Mean squared error on the boston test data after 50 iterations of training.

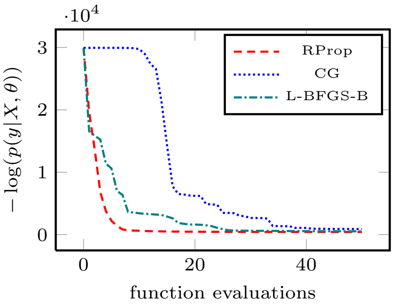

Fig. 6: Log-likelihood training curve using the CO_2 dataset.

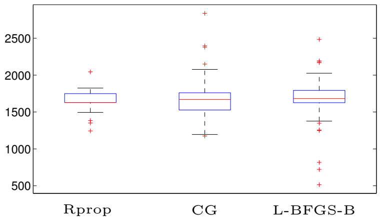

Fig. 7: Mean squared error on the CO_2 test data after 50 iterations of training.

data using a squared exponential covariance function with automatic relevance detection, which leads to a 15-dimensional optimization problem.

3.3 Mauna Loa CO_2 concentration

The Mauna Loa CO_2 dataset (Fig. 3) [8] contains monthly atmospheric carbon dioxide concentrations at Mauna Loa Observatory, Hawaii. The characteristic properties of the data can be described by a long term upward trend, as well as seasonal variations and irregular disturbances. We use the same composite covariance function as in [1], consisting of a squared exponential modeling the long term trend, a product of a squared exponential and a periodic covariance function modeling the seasonal variation, a rational quadratic covariance function modeling the irregular disturbances and a noise model.

4 Results

We evaluated the performance of Rprop compared to conjugate gradients and L-BFGS-B by considering the log-likelihood value with respect to the number of gradient evaluations. Since the necessary computation time is dominated by the evaluation the objective function and its gradient, this is a reasonable measure. Nevertheless, the analysis of the real computation time in Fig. 1 shows, that L-BFGS-B needs significantly longer than Rprop and conjugate gradients for 25 function evaluations. This computational overhead pays off while training on synthetic data, where L-BFGS-B converges after about only 5 iterations while Rprop needs almost twice as much.

On the Boston housing data set L-BFGS-B is also able to quickly improve the results, but is superseded by Rprop after 7 iterations (Fig. 4). On the CO_2 data the results are even more articulated and Rprop is able to find the optimum much faster that L-BFGS-B (Fig. 6). Conjugate gradients perform worst in all three experiments. Fig. 5 and 7 show the mean squared errors on the test data after 50 iterations. Note that for CG and L-BFGS-B the error spreads wider, suggesting convergence to suboptimal solutions.

5 Conclusion

We showed that Rprop can be successfully used for hyperparameter optimization of Gaussian processes. The performance on real world data is superior to more elaborate methods, like conjugate gradients or L-BFGS-B, while it is much easier to implement. A `C++` implementation of GP training using our method is freely available for download[1].

References

[1] Carl Edward Rasmussen and Christopher K.I. Williams. *Gaussian processes for machine learning.* The MIT Press, Cambridge, MA, April 2006.

[2] Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: the RPROP algorithm. In *IEEE International Conference on Neural Networks*, pages 586–591, San Francisco, 1993.

[3] Martin Lauer, Sascha Lange, and Martin Riedmiller. Calculating the Perfect Match: an Efficient and Accurate Approach for Robot Self-Localization. *Robocup 2005: Robot Soccer World Cup IX*, pages 142–153, 2006.

[4] Boris Lau, Christoph Sprunk, and Wolfram Burgard. Kinodynamic motion planning for mobile robots using splines. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2427–2433, October 2009.

[5] Apostolos Kotsialos, Markos Papageorgiou, Morgan Mangeas, and Habib Haj-Salem. Coordinated and integrated control of motorway networks via non-linear optimal control. *Transportation Research Part C: Emerging Technologies*, 10(1):65–84, February 2002.

[6] Carl Edward Rasmussen and H Nickisch. Gaussian Processes for Machine Learning (GPML) Toolbox. *The Journal of Machine Learning Research*, 11:3011–3015, 2010.

[7] Richard H Byrd, P Lu, J Nocedal, and C Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1994.

[8] R.F. Keeling, S.C. Piper, A.F. Bollenbacher, and J.S. Walker. Atmospheric CO2 records from sites in the SIO air sampling network. *In Trends: A Compendium of Data on Global Change*, 2009.

[1] `http://ml.informatik.uni-freiburg.de/research/libgp`